SAVONIA

OPINNÄYTETYÖ - AMMATTIKORKEAKOULUTUTKINTO
TEKNIIKAN ALA

TYÖNOHJAUSJÄRJESTELMÄN KEHITYS: PROSESSIN ASIAK-KUUSKOHTAINEN LAAJENNUS JA DOKUMENTOINTIKÄYTÄNNÖT

Opinnäytetyö

TEKIJÄ Riku Ovaskainen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma	
Työn tekijä Riku Ovaskainen	
Työn nimi Työnohjausjärjestelmän kehitys: Prosessin asiakkuuskohtainen laajennus ja dokumentointikäytännöt	
Päiväys 11.3.2025	32/1
Yhteistyötaho Eltel Networks oy	

Opinnäytetyön tilaajana toimi Eltel Networks ja sen tarkoituksena oli laajentaa ja kehittää nykyistä projektinhallinnan prosessimallia asiakkuuskohtaiseksi kokonaisuudeksi sekä kehittää nykyisiä dokumentointikäytäntöjä vastaamaan paremmin liiketoiminnan tarpeita ja tulevia kehitysprojekteja. Aihe oli syntynyt tarpeesta lisätä dokumentaation määrää ja laatua sekä tarpeesta kuvata liiketoiminnan sekä järjestelmien eri prosesseja laajemmin. Olemassa oleva dokumentaatio oli hajallaan organisaation sisällä ja tällä pyrittiin luomaan dokumentaatiolle yhtenäisempi käytäntö.

Työssäni laadittiin laajennettu malli prosessien kuvaamiseen yhdelle asiakkuudelle ja mallipohja kehityksen aikaiselle dokumentoinnille. Aineisto kerättiin dokumenttianalyysin, haastattelujen, havainnoinnin ja kokeilevan kehittämisen avulla. Työssä koottiin yhteen asiakkuuden erityisvaatimukset, järjestelmäriippuvuudet ja keskeiset prosessit ylätasolla mikä mahdollisti kokonaisuuden paremman ymmärtämisen.

Tulosten perusteella dokumentointia syntyi aiempaa enemmän ja se oli myös yhtenäisempää ja paremmin saatavilla. Johtopäätöksenä voidaan todeta, että prosessien ylätason kuvaus ja rakenteinen dokumentointimalli tukivat järjestelmäkehitystä ja lisäsivät kaivattua läpinäkyvyyttä kehitykseen. Lisäksi työ osoitti tarpeen vahvistaa dokumentaation omistajuutta ja päivitystä, mikä tarjosi selkeän jatkokehityskohteen.

Avainsanat
Toiminnanohjausjärjestelmä, työnohjausjärjestelmä, mallintaminen, projektinhallinta

SISÄLTÖ

1 JOHDANTO 4

2 KÄSITTEET 5

3 PROJEKTINHALLINTA JA KEHITYSPROSESSIEN DOKUMENTOINTI 6

3.1 Projektinhallinta 6

3.2 Projektin elinkaari 7

3.3 Projektinhallinnan menetelmät 8

3.3.1 Vesiputousmalli 8

3.3.2 Ketterät menetelmät 9

3.3.3 Lean ja kanban 11

3.4 Työnkulkujen ja prosessien mallintaminen 11

3.4.1 Mallintamisen vaiheet 12

3.4.2 Erilaiset prosessit 12

3.4.3 Menetelmät ja standardit 13

3.5 Järjestelmäkehityksen dokumentointikäytännöt 13

3.5.1 Dokumentointi ketterässä kehityksessä 14

3.5.2 Dokumentaatiotyypit 15

3.5.3 Parhaat käytännöt 16

4 TIETOJÄRJESTELMÄYMPÄRISTÖ JA KOKONAISARKKITEHTUURI 18

4.1 IFS - toiminnanohjausjärjestelmä 18

4.2 MWF – työnohjausjärjestelmä 19

4.3 Frends – integraatioalusta 20

5 TYÖN TOTEUTUS JA KEHITYSPROSESSI 21

5.1 Tutkimusote 21

5.2 Menetelmät ja aineiston keruu 21

5.3 Lähtötilanne ja suunnittelu 21

5.4 Tavoitetila ja ratkaisut 23

5.5 Projektinhallintamallin toteutus työnohjausjärjestelmään 25

6 YHTEENVETO JA TULOKSET 26

7 POHDINTA 27

LÄHTEET 28

LIITE 1: DOKUMENTAATIOPOHJA 31

1 JOHDANTO

Projektinhallinnalla on nykyään keskeinen merkitys liiketoiminnassa. Organisaatiot toimivat yhä monimutkaisemmassa ja nopeasti muuttuvassa ympäristössä, jossa projektitoiminnan avulla kehitetään uusia palveluja, toteutetaan strategisia tavoitteita ja vastataan asiakkaiden kasvaviin vaatimuksiin. Hyvin suunniteltu ja johdettu projekti varmistaa, että tavoitteet saavutetaan aikataulussa, budjetissa ja riittävän laadukkaasti. Projektinhallinta ei ole pelkkää tehtävien hallintaa vaan kokonaisvaltainen prosessi, joka kattaa suunnittelun, toteutuksen, seurannan ja arvioinnin. Sen avulla voidaan myös hallita riskejä, varmistaa resurssien tehokas käyttö ja luoda läpinäkyvyyttä organisaation toimintaan. (Tieturi – opas onnistuneeseen projektinjohtamiseen.)

Tämän opinnäytetyön tarkoituksena on laajentaa Eltelin projektinhallinnan prosessimallia yhden asiakkuuden osalta sekä kuvata projektimuotoisen työn kulku tietojärjestelmissä. Lisäksi tarkoituksena on tuottaa dokumentointimalli ja ohjeistus, joka tukee projektinhallintamallin jalkauttamista MWF-järjestelmään.

Opinnäytetyön tavoitteena on lisätä dokumentaation määrää ja parantaa sen laatua kehitysprojekteissa, sekä lisätä projektinhallinnan läpinäkyvyyttä, yhtenäisyyttä ja seurantaa. Opinnäytetyö tukee Eltelin työnohjausjärjestelmän kehitystä siten, että projektimuotoista työtä voidaan johtaa yhdenmukaisesti ja tehokkaasti tietojärjestelmien kautta.

Toimin sähköliiketoiminnan kehityskoordinaattorina, joten minun roolini on toimia kehitysprojekteissa yhdistävänä tekijänä asiantuntijoiden välillä sekä ylläpitää kehitysympäristöä. Sen vuoksi varsinainen tekninen toteutus tapahtuu asiantuntijoiden toimesta eikä siksi ole tämän opinnäytetyön keskiössä. Mallin integrointi osaksi työnohjausjärjestelmää kuvataan ylätasolta siltä osin kuin se liittyy mallinnukseen ja dokumentaatioon. Tämä työ suoritetaan osana suurempaa kehitysprojektia, jossa tarkoituksena on kehittää olemassa olevaa MWF työnohjausjärjestelmää tukemaan paremmin sähköliiketoiminnan tarpeita ja vastaamaan työnjohdollisiin tarpeisiin projektiliiketoiminnan näkökulmasta. Tämä opinnäytetyö tulee vahvasti tukemaan myös tulevia kehityshankkeita.

Työn tilaajana toimii Eltel Networks, joka toimii kriittisen infrastruktuurin, sähkön ja tietoliikenteen alalla johtavana palveluntuottajana. Projektinhallinnalla on tässä toimintaympäristössä merkittävä osuus taloudellisen ja operatiivisen onnistumisen kannalta Eltel toteuttaa laaja-alaisia hankkeita pienistä verkostonrakennustöistä suuriin investointiprojekteihin, mikä tuo omat haasteensa projektienhallinnan suhteen.

Eltelillä on käytössä toiminnanohjausjärjestelmä IFS, integraatioalusta FRENDS sekä työnohjausjärjestelmä MWF ja sen käyttöliittymät Planner, Elmo, Mobile ja Cockpit. Tässä opinnäytetyössä tarkastellaan muutoksia, kehitystä ja dokumentointia erityisesti IFS, FRENDS sekä Planner järjestelmien osalta.

Työ suoritetaan laadullisena kehittämistutkimuksena. Aineistoa kerätään haastatteluina, havainnoinnilla sekä organisaation sisäistä dokumentaatiota hyödyntämällä. Tietoturvan ja luottamuksellisuuden varmistamiseksi järjestelmätason yksityiskohtiin tai arkaluontoisiin prosessikohtaisiin tietoihin ei tässä opinnäytetyössä syvennytä vaan toimintoja ja muutoksia tarkastellaan yleisluontoisesti.

2 KÄSITTEET

API	Application Programming Interface, ohjelmistorajapinta (Junnila 2024.)
ERP	Enterprise Resource Planning, toiminnanohjausjärjestelmä (Hoffman 2024.)
FRENDS	Integraatioalusta (Junnila 2024.)
IFS	Industrial and Financial System AB, Toiminnanohjausjärjestelmä (finder.fi, nd.)
Ketterä dokumentointi	Tapa dokumentoida juuri riittävästi ja mahdollisimman kevyesti (Agile modeling nd.)
Ketterä kehitys	Ketterä kehitys on ajatusmalli, jonka ominaispiirteitä ovat iteratiivisuus, yhteistyö ja kyky sopeutua muutoksiin (koulutus.fi. 2025).
Low-code	Tapa kehittää sovelluksia käyttäen uudelleenkäytettäviä komponentteja ilman tarvetta syvälliselle ohjelmointiosaamiselle (Rokis & Kirikova 2023.)
Microsoft Visio	Ohjelmisto mallintamiseen ja kaavioiden luomiseen (Microsoft nd.)
MWF	Mobile Workforce Environment, Työnohjausjärjestelmä (Atlassian Eltel wiki.)
Ohjelmistointegraatio	Ohjelmistojen ja järjestelmien liittäminen yhteen (Junnila 2024.)
Projektinhallinta	Järjestelmällinen tapa suunnitella, toteuttaa ja seurata projekteja niin, että tavoitteet, aikataulu, budjetti ja resurssit pysyvät hallinnassa (Projektinhallinnan perusteet: Kattava opas onnistuneeseen projektijohtamiseen. Mikä on projektinhallinta 2025).

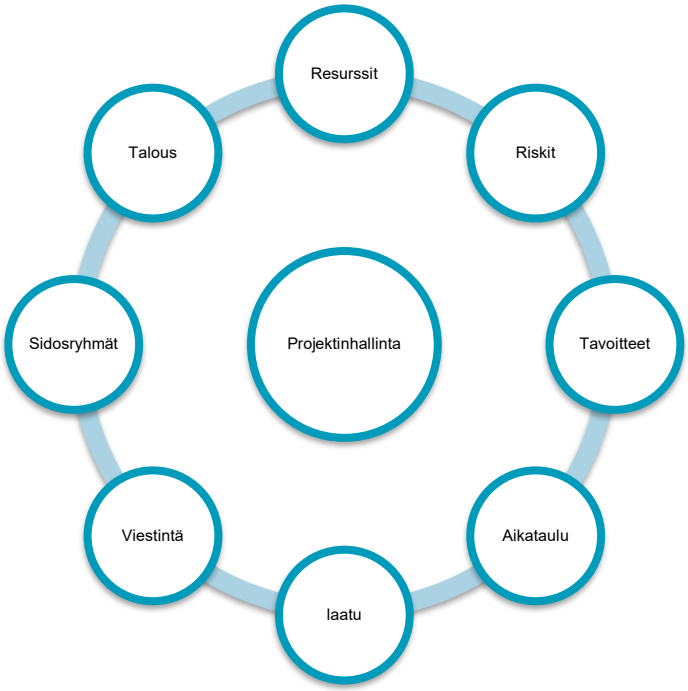
3 PROJEKTINHALLINTA JA KEHITYSPROSESSIEN DOKUMENTOINTI

3.1 Projektinhallinta

Projektinhallinta tarkoittaa järjestelmällistä tapaa suunnitella, toteuttaa ja seurata projekteja niin, että tavoitteet, aikataulu ja resurssit pysyvät hallinnassa (Tieturi 2025). Projektilla on aina rajattu alku ja loppu sekä selkeä tavoite ja sen toteuttamiseen muodostetaan yleensä määräaikainen projektiorganisaatio (Mäntyneva 2025, 7).

Projektien avulla voidaan toteuttaa muutoksia, kehittää uusia tuotteita ja palveluja sekä vastata nopeasti markkinoiden ja asiakkaiden muuttuviin tarpeisiin. Projektimaisen työskentelyn tavoitteena on yleensä parantaa toiminnan kannattavuutta, alentaa kustannuksia sekä parantaa työsuoritusten laatua, sekä mitattavuutta (Artto ym. 2008, 21-22; Mäntyneva 2025 7.)

Hyvin johdettu projekti edellyttää selkeitä käytäntöjä ja ymmärrystä projektinhallinnan keskeisistä periaatteista. Tehokas viestintä, osaava henkilöstö sekä hyvä riskienhallinta edistävät projektin taloudellista onnistumista (Artto ym. 2020.) Artto ym. (2020, 37) mukaan projektinhallinnan keskeisiä osa-alueita ovat tavoitteiden määrittely, riskienhallinta, resurssien koordinointi, aikataulun, laadun, viestinnän ja kustannusten hallinta. Sidosryhmien odotusten tasapainottaminen on myös olennainen osa projektin johtamista sillä ne voivat olla keskenään ristiriitaisia tai muuttua projektin aikana (Artto ym. 2020, 38–39). Kuvassa 1 on esitetty projektinhallinnan keskeisimmät osa-alueet.



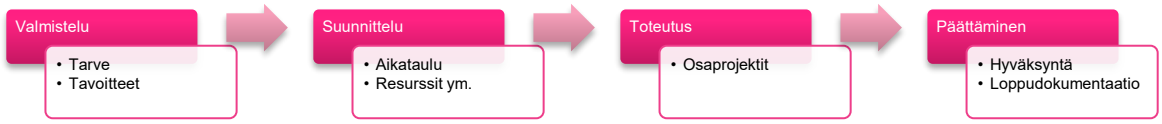
Kuva 1. Projektinhallinnan osa-alueet

Projektien luonne vaihtelee toimialoittain. Sähköurakoinnin kaltaisissa toimitusprojekteissa aikataulun ja kustannusten hallinta ovat kriittisessä roolissa, ja yhteistyö tilaajan kanssa perustuu yhteiseen projektisuunnitelmaan, vastuunjakoon sekä usein yhteiseen projektiryhmään (Mäntyneva 2025, 19-20). Toimitusprojekteissa projektimallin ennustettavuus on tärkeää, kun taas sisäisissä kehitysprojekteissa voidaan hyödyntää ketterämpää, iteratiivista lähestymistapaa toiminnan kehittämiseksi (Artto ym. 2020, 20-21; Mäntyneva ym. 2025, 7).

Projektien taustalla on aina tarpeita, kuten toiminnan tehostaminen, prosessien kehittäminen tai asiakasvaatimuksiin vastaaminen (Mäntyneva 2025, 7). Kehitysprojekti voi tuottaa lisäarvoa esimerkiksi parantamalla tietojärjestelmiä, yhdenmukaistamalla toimintamalleja tai luomalla uusia palveluita (Artto ym. 2020, 21–22). Siksi projektinhallinta kytkeytyy suoraan organisaation strategisiin tavoitteisiin, tehokkuuteen ja kykyyn tuottaa laadukkaita palveluita.

3.2 Projektin elinkaari

Mäntyneva (2025, 13) jakaa projektin elinkaaren neljään eri vaiheeseen: valmistelu, suunnittelu, toteutus ja päättäminen. Elinkaaren jäsentäminen auttaa hahmottamaan projektien kokonaisuutta ja tunnistamaan sen kannalta kriittiset työvaiheet (kuva 2).



Kuva 2. Projektin elinkaaren vaiheet

Valmisteluvaiheessa (kuva 2) määritellään projektin tarve, laajuus ja tavoitteet. Tähän vaiheeseen voi liittyä ideointi, lähtötilanteen sekä esimerkiksi tarjouslaskenta (Artto ym. 2020, 47). Kaikki projektit eivät välttämättä koskaan etene tätä vaihetta pidemmälle ja onkin mahdollista, että osa projekteista jää kokonaan toetutumatta. Huolellinen alkuvaihe luo kuitenkin paremman perustan ja luo edellytykset onnistuneelle suunnittelulle (Mäntyneva 2025, 13.)

Suunnitteluvaiheessa (kuva 2) täsmennetään projektin tavoitteet, aikataulu ja resurssit sekä arvioidaan toteutusvaihtoehtoja. Tässä vaiheessa muodostetaan projektiorganisaatio ja laaditaan projektisuunnitelma, joka sisältää muun muassa taustan, vastuut, budjetin ja viestinnän periaatteet projektin aikana (Artto ym. 2020, 52.) Toimitusprojekteissa on tärkeää huomioida myös asiakkaan tavoitteet, sillä onnistuminen vahvistaa asiakassuhdetta ja vähentää ristiriitojen mahdollisuutta. Nykyisin ketterissä projektinhallintamenetelmissä liian tarkkaa ja huolellista pitkän aikavälin suunnittelua ei pidetä järkevänä, vaan tavoitteena on määritellä lyhyemmillä aikaväleillä kohdennetumpia ja tarkempia toimenpiteitä projektin edistämiseksi (Mäntyneva 2025, 42.)

Toteutusvaiheessa (kuva 2) toteutetaan suunnitteluvaiheen työvaiheet ja tavoitteet. Toteutus voi koostua useammasta osiosta tai osaprojektista, jotka yhdessä muodostavat projektin lopputuloksen. Toteutusvaiheen rakenne voidaan määritellä projektin tyypin ja laajuuden mukaan. Suurissa projekteissa voi olla perusteltua pilkkoa projekti pienemmiksi osaprojekteiksi tai tehtäviksi, jolloin hallinta ja projektin seuranta helpottuu. Lisäksi osatuloutuksen hyödyntäminen mahdollistaa kassavirran paranemisen sitä mukaa kun osaprojektit valmistuvat (Mäntyneva 2025, 15-17; 59.)

Projektin päättäminen (kuva 2) edellyttää aina tilaajan hyväksymistä projektin lopputuloksen osalta. Yleensä projektin päättyminen tiedetään etukäteen ja on yleistä, että loppuvaiheessa tulee kiire saada projekti määräaikaan mennessä valmiiksi. Yleensä toimittajan viimeiset saatavat on myös sidottu valmistumiseen jollakin tavalla. Tällä tavoin saadaan toimittaja sitoutumaan ennalta määritel-

tyyn aikatauluun. Päätösvaiheessa luovutetaan työn tilaajalle projektin dokumentaatio. Dokumentaatio voi sisältää esimerkiksi projektisuunnitelman, käyttöohjeet tai erilaisia seurantaraportteja (Mäntyneva 2025, 148.)

3.3 Projektinhallinnan menetelmät

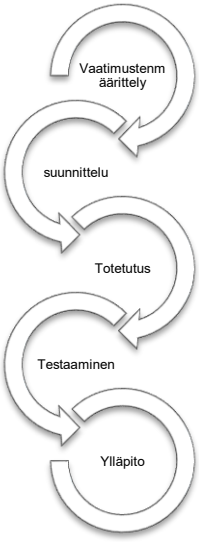
Projektinhallintaa voidaan tarkastella sen menetelmien, teknisten välineiden tai dokumentaation kautta. Tämä tarkoittaa, että projektinhallinnan tueksi on määritetty erilaisia toimintatapoja, ohjeistuksia sekä teknisiä välineitä, joita käyttämällä pyritään paremmin hallitsemaan erilaisia epävarmuuksia ja varmistamaan projektin tavoitteiden saavuttaminen (Artto ym. 2020, 40.) Nykyisin käytetään paljon erilaisia järjestelmiä projektien seurantaan, resursointiin ja raportointiin, mikä mahdollistaa paremman hallinnan, seurattavuuden ja tiedon läpinäkyvyyden projektissa.

Projekteihin voi liittyä paljon epävarmuustekijöitä ja erilaisia muuttujia, joita on vaikea ennustaa etukäteen kovin hyvin. Tämän vuoksi nykyisin projektinhallinnassa suositaan ketterämpää lähestymistapaa, joka mahdollistaa sopeutumisen ja toiminnan kehittämisen ja muutoksen vallitsevan tilanteen mukaiseksi (Project Management Institute 2017, 7.) Ketterät menetelmät korostavat iteratiivista kehitystä, tiivistä yhteistyötä sidosryhmien kanssa ja jatkuvaa parantamista, mikä tekee niistä erityisen hyödyllisiä ympäristöissä, joissa olosuhteet ja tarpeet muuttuvat nopeasti.

Kun valitaan projektinhallinnan menetelmää, on kiinnitettävä huomiota siihen, millaista projektia ollaan toteuttamassa. On viitteitä, että suurissa projekteissa, joissa muutoksia sallitaan vähemmän, on järkevämpää suosia enemmän perinteisempää projektinjohtamisen menetelmää. Tällaisia voivat olla suuret toimitusprojektit, joissa on tarkasti määritetty valmiit toimintamallit ja vaatimukset projektin toimittajalle (Project Management Institute 2017, 20.) Ketterät menetelmät sopivat paremmin nopeatempoisiin ympäristöihin, joissa tehdään paljon yhteistyötä eri sidosryhmien kanssa ja vaatimukset voivat muuttua nopeammin. Tällaisia voivat olla esimerkiksi sisäiset kehitysprojektit (Project Management Institute 2017, 24-25.) Toisaalta projekteissa voidaan käyttää myös jonkinlaista hybridimallia, jolloin voidaan hyödyntää molempien menetelmien parhaiten sopivia piirteitä. Niissä projektin vaiheissa, joissa on suurta epävarmuutta, voi olla järkevää käyttää ketterien menetelmiä ja sellaisissa vaiheissa, joissa on enemmän varmuutta vaatimusten suhteen, voidaan taas käyttää perinteisempiä projektinhallinnan menetelmiä (Project Management Institute 2017, 18-20.)

3.3.1 Vesiputousmalli

Perinteisessä vesiputousmallissa edetään lineaarisesti vaiheesta toiseen. Vaiheita voi olla erilaisia toimialasta ja toimintaympäristöstä riippuen, mutta tyypillisesti ne ovat; vaatimustenmäärittely, suunnittelu, toteutus, testaaminen ja ylläpito (Saravanos 2025.)



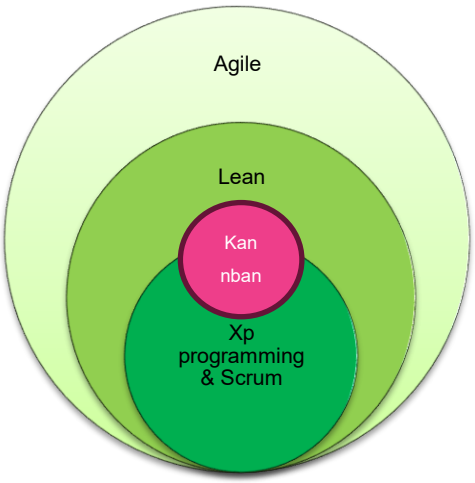
Kuva 3. Perinteinen vesiputousmalli

Vesiputousmallille (kuva 3) on ominaista, että keskitytään vain yhteen vaiheeseen kerrallaan, eikä edellisiin vaiheisiin enää palata. Tämä tekee virheiden korjaamisesta myöhemmin haastavaa. Vesiputousmallissa korostuu huolellinen suunnittelu projektin alkuvaiheissa. Sen hyötyjä ovat rakenteen selkeys sekä hyvä ennustettavuus. Varsinkin toimitusprojekteissa tämä on tilaajan kannalta helpompaa, kun voidaan tehdä tarkempia suunnitelmia ja budjetointia. Toisaalta rakenteen jäykkyys voi haitata virheiden havaitsemista ja reagointia. Muokattu vesiputousmalli voi tuoda joustavuutta perinteiseen vesiputousmalliin verrattuna mutta se ei ratkaise täysin nopeiden muutosten ja tiiviin asiakasyhteistyön haasteita (Adetokunbo, Adenowo, Basirat, Adenowo 2013.) Tähän haasteeseen vastaa paremmin erilaiset ketterät menetelmät.

3.3.2 Ketterät menetelmät

Ketterä kehittäminen on ennen kaikkea ajattelutapa kuin yksittäinen menetelmä. Siinä korostuvat iteratiivinen eteneminen, tiivis yhteistyö ja kyky reagoida muutoksiin. Agile manifesto määrittelee ketteryyden keskeiset arvot: yhteistyö ihmisten välillä, toimiva ratkaisu kattavan dokumentoinnin sijaan, asiakkaan jatkuva osallistuminen sekä muutoksiin mukautuminen alkuperäisen suunnitelman sijaan (Agile manifesto 2001; Project Management Institute 2017, 8-9.) Ketteryyttä tukevat myös kaksitoista periaatetta, joiden tavoitteena on tuottaa arvoa varhain, edistää jatkuvaa parantamista ja varmistaa sidosryhmien tiivis yhteistyö.

Ketterät menetelmät (kuva 4) sisältävät useita eri viitekehyksiä, joista Scrum painottaa sprinttipohjaista kehitystä, Kanban virtaustehokkuutta, Lean hukan poistamista ja jatkuvaa parantamista ja XP programming teknistä laatua. Usein organisaatiot yhdistelevät näitä omien tarpeiden ja osaamisen mukaisesti (Mark ym. 2020, 86-87.)



Kuva 4. Ketterän kehityksen menetelmät

Scrum on yksi käytetyimmistä ketteristä viitekehyksistä (kuva 4), ja sitä sovelletaan ohjelmistokehityksen lisäksi myös liiketoiminnan prosessien kehittämiseen. (koulutus.fi 2025). Scrum perustuu empirismiin, jossa päätöksenteko nojaa havaintoihin ja toistuvaan tarkasteluun. Sen kolme kulmakiveä ovat läpinäkyvyys, tarkastelu ja sopeutuminen, joiden avulla pyritään minimoimaan virheet ja parantamaan ennustettavuutta (Schwaber & Sutherland 2017.)

Scrum organisaatioroolit muodostuvat tuoteomistajasta, Scrum masterista ja kehitystiimistä. Tuoteomistaja vastaa arvon maksimoimisesta ja kehityskohteiden priorisoinnista, kun taas Scrum master varmistaa prosessin esteettömän etenemisen ja tukee tiimin oppimista. Kehitystiimi toteuttaa sovitut tehtävät sprinttien aikana (Schwaber & Sutherland 2017.)

Kehitys etenee lyhyissä, tyypillisesti enintään kuukauden mittaisissa sprinteissä, joiden lopputuloksena syntyy valmis inkrementti. Sprinteissä hyödynnetään erilaisia tapahtumia, kuten suunnittelua, katselmointia ja retrospektiivejä. Keskeiset Scrum-artefaktit ovat tuotteen backlog, sprint backlog sekä inkrementti, jotka jäsentävät työn etenemistä ja parantavat läpinäkyvyyttä (Schwaber & Sutherland 2017.)

XP programming ja Scrum viitekehys ovat monella tavalla samankaltaisia. Molemmat korostavat iteratiivista kehitystä, tiivistä yhteistyötä, asiakaspalautetta sekä asiakasarvon tuottamista. Keskeisin ero niiden välillä liittyy teknisiin käytäntöihin. XP painottaa ohjelmiston teknisen laadun parantamista muun muassa pariohjelmoinnin, testivetoisen kehityksen (test driven development) sekä lyhyempien iteraatioiden avulla (Schwaber & Sutherland 2017; Mark ym. 2020, 97-98.)

Standish Groupin vuonna 2015 tekemän tutkimuksen mukaan vain 29 prosenttia ohjelmistoprojekteista oli onnistuneita ottaen huomioon aikataulun, budjetin ja lopputuloksen. Tutkimuksesta selviää, että projektit, joissa käytettiin perinteistä vesiputousmenetelmää, luokiteltiin onnistuneiksi 11 prosentissa tapauksista ja ketteriä menetelmiä käytettäessä prosentti nousi 39 prosenttiin. Tutkimuksen mukaan myös projektin koolla on väliä. Suuremmat projektit usein epäonnistuvat todennäköisemmin kuin pienet, huolimatta siitä mitä menetelmää käytetään (Standish Group 2015.)

3.3.3 Lean ja kanban

Siinä missä ketterät menetelmät painottavat joustavuutta ja nopeaa reagointia muutoksiin, Lean menetelmä painottaa prosessin virtaviivaistamista, hukan poistamista ja liiketoiminta-arvon maksimointia. Hukan minimoiminen tarkoittaa, että prosessista poistetaan kaikki asiat, jotka eivät lisää arvoa asiakkaalle. Hukalla tarkoitetaan esimerkiksi ylimääräisiä prosessin vaiheita, tarpeettomia kokouksia, tai keskittymistä vääriin asioihin. (Poppendieck & Poppendieck 2003.)

Ennustettavuutta pyritään kasvattamaan oppimista vahvistamalla. Työ tehdään läpinäkyväksi, sitä tarkastellaan säännöllisesti ja toimintaa sopeutetaan löydösten avulla. Päätökset tehdään vasta sitten kun käytössä on mahdollisimman paljon tietoa päätöksen tueksi, tällä tavoin vältetään tekemästä oletuksia, joille ei ole riittävää tietoperustaa. Leanissa pyritään samalla tavoin, kun Scrum ja XP programming viitekehyksissä toimittamaan ratkaisuja nopeasti. Keskeneräisen työn rajoittaminen (WIP, work in progress), virtauksen optimointi ja julkaisuaikojen lyhentäminen nopeuttavat palautteen saamista, mikä puolestaan mahdollistaa kehitystyön jatkuvan parantamisen (Poppendieck & Poppendieck 2003; Mark ym. 2020 89-91.)

Leanissa korostetaan myös tiimien ja työntekijöiden itseohjautuvuutta. Esimiesten tehtävänä on poistaa työn tekemisen esteitä, antaa tekijöille tarvittavat työkalut ja edistää hyvän työympäristön luomista. Laadunhallinnan tulee olla sisäänrakennettuna. Tämä tarkoittaa, että virheet pitäisi pystyä havaitsemaan aikaisin ja vähentää mahdollisia riippuvuuksia siten että kehitystä voidaan tehdä missä vaiheessa tahansa, ilman että kehitys hidastuu. Leanissa korostetaan kokonaisuuden hahmottamista. Joissakin tapauksissa osaprosessin optimointi saattaa heikentää kokonaisprosessia, sen vuoksi kehittämistoimenpiteet on arvioitava aina koko järjestelmän näkökulmasta (Poppendieck & Poppendieck 2003; Mark ym. 2020, 89-90.)

Kanbanin avulla voidaan visualisoida prosessi tai kehitysvaiheet helposti ymmärrettäväksi kokonaisuudeksi. Kanban metodissa ei edellytä iteraatioita, vaan työ viedään prosessin läpi virtausta optimoiden. Kanbanin näkökulmasta vain valmiilla työllä on arvoa, joten työn virtausta priorisoidaan. Uutta työtä vai vaihetta ei aloiteta ennen kuin siihen tarvittava kapasiteetti on olemassa. Tämä vähentää keskeneräisen työn määrää, hukkaa ja vaihtelun aiheuttamaa häiriötä. Kanbanin avulla voidaan tunnistaa pullonkauloja ja hidasteita kehityksessä visualisoimalla työvaiheita ja työnkulkua (Mark ym. 2020, 103-105.) Kanban ja Lean kytkeytyvät siis vahvasti toisiinsa ja Kanban voidaan nähdä Leanin yhtenä työkaluna.

3.4 Työnkulkujen ja prosessien mallintaminen

Edellisisissä luvuissa tarkasteltiin järjestelmäkehityksen menetelmiä ja periaatteita sekä niiden merkitystä organisaation kehitystyössä. Erityisesti Lean ajattelussa korostettiin prosessien virtaviivaistamista ja hukan poistamista yrityksen tuloksellisuuden parantamisessa. Lean ajattelu luo vahvan perustan myös työnkulkujen ja erilaisten liiketoiminnan prosessien mallintamiselle. Mallintaminen auttaa hahmottamaan nykytilan prosessit, tunnistamaan kehityskohteet ja varmistamaan että järjestelmäkehitys tukee liiketoiminnan tavoitteita. Selkeä prosessimalli parantaa ymmärrystä eri sidosryhmien välillä, vähentää virheitä ja toimii dokumentaation pohjana.

3.4.1 Mallintamisen vaiheet

Prosessiajatteluun liittyy vahvasti arvon tuottaminen sekä systemaattinen prosessin parantaminen, toiminnan tunnistaminen ja kehitys prosessikuvausten avulla. Prosessien mallintamista käytetään usein tietojärjestelmien sekä liiketoiminnan prosessien kehityksessä. Ennen mallintamista on mietittävä tarkasti rajaus sekä kuvauksen tarkkuus. Esimerkiksi järjestelmäkehityksessä hyvin yksityiskohtainen kuvaus vanhentuu nopeasti, jos muutoksia tehdään nopealla tahdilla. Tällainen kuvaus voi olla tarpeen asiantuntijoille mutta esimerkiksi liiketoimintajohdolle riittää usein ylätason kuvaus, jossa ei mennä liikaa teknisiin yksityiskohtiin. Tällaisissa kuvauksissa mallinnetaan enemmän logiikkaa kuin tarkkoja teknisiä toimintoja. Prosesseilla voi olla monta tasoa ja ne voi jakautua pää -ali tai osaprosesseiksi, joten kuvauksia voi olla tarpeen tehdä useampia siten että tarkoitus ja yksityiskohdat selviävät riittävällä tarkkuudella (Martinsuo, Blomqvist 2010.)

Prosessin mallintaminen aloitetaan yleensä nykytilanteen kuvaamisella. Prosessi pyritään rajaamaan mahdollisimman tarkasti saatavilla olevan tiedon perusteella. Tähän voidaan käyttää monenlaisia menetelmiä kuten haastatteluita, tietokanta analyysiä tai simulointia. Kuvaamisen aikana tai sen jälkeen huomataan usein kehityskohteita. Nykytilanteen kuvaamisen jälkeen prosessia verrataan tahtotilaan, eli siihen miten prosessin pitäisi toimia. Tämä voi vaatia muutoksia nykyiseen prosessiin tai tarvittaessa suurempaan arkkitehtuuriseen kokonaisuuteen. Riippuen siitä millainen prosessi on kyseessä, tällä voidaan tarkoittaa esimerkiksi toimintatapojen muutosta, muutosta järjestelmätasolla tai ohjeistuksissa (Martinsuo, Blomqvist 2010.)

Kun nykytilanne ja tavoiteprosessi on määritelty, aloitetaan testausvaihe. Tavoiteltu prosessi kannattaa testata hallitusti ja sitä pitää myös seurata, että jokainen prosessin vaihe toimii odotetulla tavalla. Tarvittaessa voidaan tehdä viimeisiä muutoksia ennen varsinaista jalkauttamista. Tässä vaiheessa myös todennetaan mitä lisäarvoa uusi prosessi mahdollisesti loi ja saatiinko sillä ratkaistua olemassa olevia ongelmia. Pilotoinnin jälkeen on myös tarpeen seurata, että prosessi toimii odotusten mukaisesti (Martinsuo, Blomqvist 2010.)

3.4.2 Erilaiset prosessit

Liiketoimintaprosesseista puhutaan yleensä silloin, kun kyseessä on prosessi, jonka avulla yritys tekee rahaa (Martinsuo, Blomqvist 2010). Englanninkielinen vastine Business process management tarkoittaa liiketoiminnan prosessien hallintaa. Johan Nelis ja John Jeston (Busines Process Management) määrittelee käsitteen vapaasti suomennettuna seuraavasti; BPM on johtamisen osa-alue, joka keskittyy hyödyntämään liiketoimintaprosesseja organisaation tavoitteiden saavuttamisessa. Se sisältää prosessien jatkuvan parantamisen sekä suorituskyvyn hallinnan ja ohjauksen, jotta keskeiset prosessit tukevat strategisia päämääriä. BPM hallintaan on kehitetty useita hyödyllisiä työkaluja, mutta keskiössä pitäisi teknologian lisäksi olla prosessien jatkuva parantaminen sekä tehokkuus, jotka voidaan useimmiten saavuttaa myös ilman teknologiaa. Prosesseja voidaan tehostaa lisäksi automatisoimalla erilaisia työvaiheita mikä tehostaa prosessin virtausta ja parantaa laatua (Nelis, Jeston 2010, 56-68.) Liiketoiminnan prosessien hallinnassa on kyse liiketoiminnan tarpeista lähtevistä kuvauksista, jotka usein ovat teknisesti suppeita ja keskittyvät loogisiin kokonaisuuksiin.

Siinä missä liiketoimintaprosessit kuvaavat organisaation toiminnan kokonaisuuksia, tekniset prosessit liittyvät järjestelmien sisäisiin toimintoihin. Käytännössä asiaa voidaan ajatella siten että liike-

toimintaprosessi kertoo mitä tehdään, ja tekninen prosessi kertoo miten asia pitää toteuttaa. Kehitysprojektit lähtevät yleensä liikkeelle liiketoimintaprosessien kehitystarpeista, jotka sitten käännetään teknisiksi prosesseiksi.

3.4.3 Menetelmät ja standardit

Prosessien ja järjestelmien mallintamisessa käytetään useita erilaisia menetelmiä ja standardeja, jotka palvelevat eri tarkoituksia. Oikean menetelmän valinta riippuu siitä, kuvataanko liiketoiminnan prosesseja, ohjelmiston rakenteita vai järjestelmän kokonaisarkkitehtuuria. Kaksi yleisintä standardia ovat BPMN ja UML, joiden ylläpidosta vastaa nykyisin Object Management Group (OMG).

Business Process Management Notation (BPMN) on kansainvälinen standardi liiketoimintaprosessien mallinnukseen. Sen avulla prosessit voidaan kuvata selkeästi ja ymmärrettävästi erilaisten vuokaavioiden avulla. BPMN:n keskeinen tavoite on luoda helposti ymmärrettävä merkintätapa kaikille prosessien parissa työskenteleville, roolista riippumatta. Standardoitua merkintätapaa voidaan hyödyntää prosessin jokaisessa vaiheessa suunnittelusta tekniseen toetutukseen. Merkintätapa on riittävän yksinkertainen liiketoiminnan tarpeisiin, mutta samalla tarkka, jotta kaaviot voidaan muuntaa tarvittaessa ohjelmistoprosessien komponenteiksi. BPMN tukee myös automaatiota ja varmistaa yhteensopivuuden XML-pohjaisten kielten, kuten WSBPEL:n kanssa. BPMN on vakiintunut standardi ja sen avulla voidaan parantaa myös dokumentaatiota ja ymmärrystä asiantuntijoiden ja liiketoiminnan välillä (Graphical notations for business processes nd.)

Unified modelling language (UML) on toinen suosittu visuaalinen mallinnustapa prosessien sekä erilaisten monimutkaisten järjestelmien mallintamiseen. Sitä käytetään usein olio ja komponenttipohjaisten järjestelmien rakentamisessa. Siitä huolimatta se ei kuitenkaan ole sidoksissa ohjelmointikieliin tai kehitysprosesseihin vaan sen tarkoitus on olla yleiskäyttöinen visuaalinen työkalu, jolla voidaan kehittää, dokumentoida ja jakaa erilaisia järjestelmäarkkitehtuureja ja ohjelmistomalleja (Graphical notations for business processes nd.)

Ylätason kuvaukset ilman liian tarkkaa teknistä yksityiskohtaisuutta sopii strategiseen suunnitteluun ja kommunikointiin liiketoimintajohdon kanssa, kun taas UML ja BPMN standardin mukaiset vuokaaviot helpottavat yksityiskohtaisten prosessien kuvaamista, jossa mallinnetaan prosessin etenemistä alkupisteestä loppupisteeseen. Lisäksi ne sopivat hyvin erilaisten tietovirtojen kuvaamiseen joiden avulla voidaan tunnistaa erilaisia pullonkauloja prosesseissa tai suunnitella automaatiota. Erilaisten uimarata kaavioiden avulla voidaan helposti kuvata työnkulkuja ja vastuita sekä erilaisia rooleja prosessissa. Yhdessä eri kaaviot tuottavat ymmärrettävän kuvauksen prosessista, jossa huomioidaan sekä liiketoiminnan että teknisen toteutuksen tarpeet.

3.5 Järjestelmäkehityksen dokumentointikäytännöt

Dokumentoinnilla on järjestelmäkehityksessä keskeinen rooli järjestelmien suunnittelun, toteutuksen ylläpidon ja jatkokehityksen tukena. Hyvin laadittu dokumentaatio parantaa ymmärrystä järjestelmistä, vähentää hiljaisen tiedon määrää ja tukee tiedon siirtymistä eri sidosryhmien välillä. Dokumentaatiokäytännöt ovat siten olennainen osa laadukasta ja kestävää järjestelmäkehitystä. Tässä luvussa tarkastellaan järjestelmäkehityksen dokumentointikäytäntöjä erityisesti ketterän kehityksen näkökulmasta sekä tavoitteita ja toteuttamisen periaatteita osana modernia ohjelmistokehitystä.

3.5.1 Dokumentointi ketterässä kehityksessä

Ohjelmistokehityksen dokumentointikäytännöt ovat viime vuosina muuttuneet merkittävästi ketterien menetelmien yleistymisen myötä. Jo Agile Manifestossa korostettiin toimivaa ohjelmistoa kattavaa dokumentaatiota tärkeämpänä, mikä on osaltaan haastanut perinteiset dokumentointikäytännöt. Tämä periaate on kuitenkin ajoittain tulkittu virheellisesti siten, että dokumentaation merkitystä on vähätelty tai dokumentointi on jäänyt puutteelliseksi. Todellisuudessa dokumentaatio on olennainen osa kaikkia ohjelmistoprojekteja, mutta ketterissä menetelmissä sen määrää ja sisältöä pyritään sopeuttamaan todelliseen tarpeeseen. Erityisesti kehityksen alkuvaiheessa suositellaan välttämään ylimääräistä ja raskasta dokumentaatiota, joka ei tuota välitöntä arvoa kehitystyölle (Voigt, Garrel, Müller & Wirth 2016.)

Ketterässä kehityksessä dokumentaatiota ei tule nähdä erillisenä tai kehitystä hidastavana vaiheena, vaan luontevana osana ohjelmistokehitysprosessia. Dokumentaatiota tuotetaan ja täydennetään iteratiivisesti kehitystyön edetessä. Koska jokaisella organisaatiolla ja järjestelmällä on omat erityispiirteensä, dokumentointikäytännöt tulee mukauttaa siten, että dokumentaatiota syntyy riittävästi ilman, että ketterien menetelmien tuomat hyödyt vaarantuvat (Laine 2021.)

Perinteisessä vesiputousmalliin pohjautuvassa ohjelmistokehityksessä dokumentaatioon käytetään tyypillisesti huomattavasti enemmän aikaa ja resursseja kuin ketterissä kehitysmalleissa. Projektin alkuvaiheessa laaditaan yksityiskohtainen vaatimusmäärittely, joka toimii pohjana kaikille myöhemmille työvaiheille. Tämän jälkeen tuotetaan useita erilaisia määrittely- ja suunnitteludokumentteja, joissa järjestelmän toiminta, rakenne ja tietomallit kuvataan tarkasti ennen varsinaisen toteutuksen aloittamista. Toteutusvaiheen oletetaan olevan pääosin suoraviivaista, kun kehittäjien tehtävänä on toteuttaa ennalta määritetyt vaatimukset koodiin. Käytännössä muutostarpeita syntyy kuitenkin väistämättä kehityksen edetessä. Lisäksi projektissa tuotetaan erillisiä testausdokumentteja sekä loppukäyttäjille suunnattua käyttöohjedokumentaatiota. Tämän seurauksena perinteisissä projekteissa syntyy suuri määrä dokumentaatiota, josta kokemusten mukaan vain rajallinen osa tuottaa todellista arvoa varsinaiselle kehitystyölle (Laine 2021.)

Ambler (Agile Modeling, n.d.) painottaa että dokumentaatiota tulisi tuottaa ainoastaan tarpeellinen määrä ja riittävällä tarkkuudella. Pyrkimys pitää dokumentaatio jatkuvasti täysin yhdenmukaisena lähdekoodin kanssa voi viedä huomattavasti aikaa itse ohjelmistokehittämiseltä. Tästä näkökulmasta riittävän hyvä ja ajantasainen dokumentaatio on arvokkaampaa kuin täydellinen mutta nopeasti vanheneva kokonaisuus.

Ketterissä menetelmissä korostetaan aktiivisen vuorovaikutuksen ja suoran kommunikoinnin merkitystä. Voigtin ym. (2016) tutkimuksen mukaan ohjelmistokehittäjien ajasta vain noin 12 % kuluu varsinaiseen koodin kirjoittamiseen, kun taas jopa 33 % ajasta kuluu keskusteluun ja tiedon etsimiseen. Tutkimustulokset osoittavat, että hyvin toteutettu dokumentaatio voi merkittävästi tehostaa kehitystyötä vähentämällä tiedonhakuun ja epäselvyyksien selvittämiseen kuluvaa aikaa.

Ohjelmiston julkaisua seuraava ylläpitovaihe kattaa kaikki järjestelmään tehtävät muutokset ja korjaukset julkaisun jälkeen (Cozzetti ym.). Ylläpidon merkitys on huomattava, sillä tutkimusten (Koski-

nen 2015) mukaan jopa 90 prosenttia ohjelmistoprojektin kokonaiskustannuksista syntyy ylläpitovaiheessa. Vaikka ketterissä menetelmissä painotetaan suullista viestintää ja yhteistyötä kehitystiimin sisällä, dokumentaation merkitys korostuu erityisesti ohjelmistojen pitkäaikaisessa ylläpidossa. Tämä on erityisen tärkeää tilanteissa, joissa järjestelmiä modernisoidaan tai siirretään uusille teknisille alustoille.

3.5.2 Dokumentaatiotyypit

Edellisessä osiossa käytiin jo vähän läpi ohjelmistoprojektien dokumentaatiota. Taulukossa 1 on esitetty esimerkkejä erilaisesta dokumentaatiosta.

Taulukko 1. Dokumentaatiotyypit

Tyyppi	Sisältää
Vaatimusmäärittely	Toiminnallinen ja ei toiminnallinen vaatimusmäärittely
Suunnittelu	Arkkitehtuuri suunnitelma Yksityiskohtainen suunnitelma
Tekninen dokumentaatio	Rajapintojen, tietokantojen ja ohjelmistokoodin dokumentaatiot
Käyttäjädokumentaatio	Käyttöohjeet ja koulutusmateriaalit
Testidokumentaatio	Testisuunnitelma ja testitapaukset, testitulokset
Projektidokumentaatio	Projektin aikataulutus, resurssit ja muutosten seuranta
Rajapintojen dokumentaatio	Swagger / Open Api, SDK interaktiivinen dokumentaatio
Ylläpito dokumentaatio	Vika raportit, muutosten seuranta
Julkaisudokumentaatio	Asennusdokumentaatio ja julkaisutiedotteet
Laki ja vaatimustenmukaisuus	Lisenssiehdot ja standardit
Turvallisuus dokumentaatio	Turvapolitiikat ja auditoinnit

Cozzetti, de Souza, Anquetil ja de Oliveira (2005) tekemässä tutkimuksessa selvitettiin mitä dokumentaatiota ohjelmistojen ylläpitäjät pitivät kaikkein tärkeimpänä. Tulokset vahvistivat, että lähdekoodi ja siihen liitetyt kommentit olivat ohjelmiston ymmärtämisen kannalta tärkeimmät tietolähteet ylläpitovaiheessa. Seuraavaksi merkittävimmiksi tietolähteiksi nousivat tietomallit sekä vaatimusmäärittelyt. Yllättävää oli se, että arkkitehtuurimallit sekä muut järjestelmän kokonaiskuvaa kuvaavat

dokumentit eivät osoittautuneet erityisen tärkeiksi. Tämä saattaa viitata siihen, että kyseisiä dokumentteja hyödynnetään lähinnä alkuvaiheessa kokonaiskuvan muodostamiseksi, jonka jälkeen niihin ei tarvitse enää palata.

Ambler (Agile modeling nd) mukaan ketterässä ohjelmistoprojektissa tarvittava dokumentaatio muodostuu sidosryhmien tarpeista, kuten käyttäjä, ylläpito ja operointidokumentaatiosta, sekä järjestelmien välisiä rajapintoja kuvaavista dokumenteista. Dokumentaatiota tarvitaan myös viestinnän tukena, organisaation tiedon säilyttämiseksi, auditointi ja sääntelyvaatimusten täyttämiseksi sekä kehitystyön ja ajattelun jäsentämiseksi. Dokumentointi voidaan nähdä liiketoiminnan investointina, jota tuotetaan todelliseen tarpeeseen.

Yleensä ohjelmistoprojektin alussa tehdään vaatimusmäärittely. Vaatimukset jaotellaan toiminnallisiin ja ei toiminnallisiin vaatimuksiin. Toiminnalliset vaatimukset ovat toimintoja ja tehtäviä, joista ohjelmiston tulee suoriutua. Ei toiminnalliset vaatimukset määrittelevät laatuun liittyviä asioita kuten suorituskyky, tietoturva ja skaalautuvuus (Nayeem Islam 2024). Perinteisestä määrittelystä poiketen ketterissä menetelmissä suositaan paljon nykyisin käyttäjätarinoita, jotka ovat aavistuksen kevyempiä kuin perinteiset vaatimusmäärittelyt. Käyttäjätarinan ajatus on kuvata tarvittavaa järjestelmän toiminnallisuutta käyttäjän näkökulmasta. Henri Laineen (2021) mukaan myös dokumentaation tarve pystytään kartoittamaan samalla tavalla. Tätä varten on tunnistettava erilaisia rooleja, joita käyttäjillä voisi olla. Laine (2021) käyttää esimerkkinä uutta kehittäjää, ”Uutena kehittäjänä haluan saada dokumentaatiosta nopeasti selville sen, missä toiminnallisuuden X toteuttava koodi sijaitsee, jotta voisin työskennellä nopeasti ja tehokkaasti tuhraamatta aikaani tämän lähdekoodin etsimiseen.”

Tämän jälkeen tarinoille laaditaan hyväksymiskriteerit. Tarkoituksena on, että käyttäjätarinat on tehty niin hyvin, että kaikki toiminnallisuuksien sekä testien dokumentaatio voidaan tuottaa samalla kertaa. Sen sijaan että testattaisiin yksikkötestitasolla, tehdäänkin toiminnallisia testejä, joilla saadaan kattavammin kartoitettua ohjelmiston haluttua toiminnallisuutta myös käyttäjän näkökulmasta. Tämä lähestymistapa yhdistää vaatimusmäärittelyn, dokumentoinnin ja testauksen yhtenäiseksi kokonaisuudeksi, mikä vähentää päällekkäistä työtä. Lisäksi dokumentaation laatu paranee, koska se syntyy osana kehitysprosessia eikä erillisenä vaiheena (Laine 2020.)

3.5.3 Parhaat käytännöt

Ahamed ym. (2024) tutkimuksen mukaan ohjelmistoprojektin alussa olisi tärkeää määritellä ohjelmistoprojektin tyyppi ja erityisvaatimukset koska näiden asioiden tulisi ohjata dokumentointikäytäntöjä. Tämän jälkeen organisaation tulisi keskittyä kohderyhmän tunnistamiseen, ajantasaisuuden varmistamiseen sekä oikeiden työkalujen valitsemiseen. Hyvä työkalu helpottaa dokumentaatiota tarjoamalla mallipohjia, automaatiota ja yhteistyöominaisuuksia. Automaation käyttäminen onkin suositeltavaa aina kun se on mahdollista (Nayeem Islam 2024.)

Teknisen dokumentaation tulee olla selkeää ja tiivistä. Asiat pitää pyrkiä selittämään mahdollisimman yksinkertaisesti, kuitenkaan yksinkertaistamatta liikaa. Rakenteen loogisuus ja selkeys helpottaa lukijaa ymmärtämään kontekstia ja parantaa luettavuutta. Sisältö kannattaa räätälöidä kohdeyleisön mukaan siten, että se palvelee sekä aloittelijoita että kokeneempia käyttäjiä (Nayeem Islam 2024.)

Satishin ym. (2016) mukaan dokumentoinnin yleisimmät haasteet jatkuvat läpi koko ohjelmiston elinkaaren. Keskeisimmät ongelmat liittyvät dokumentaation vanhentumiseen sekä sen puutteelliseen jäljitettävyyteen suhteessa koodiin tehtyihin muutoksiin. Lisäksi dokumentoinnin standardien puute sekä ohjelmistokehittäjien vähentynyt motivaatio dokumentointiin heikensi dokumentaation laatua ja ajantasaisuutta. Ahamed J. ym (2024) tutkimuksessa nostetaan esille myös arkkitehtuuridokumentaation heikkous nopeasti muuttuvan kehityksen aikana sekä saatavuuteen ja laatuun liittyvät ongelmat.

Hyvä dokumentaatio edellyttää vakiintuneita hyväksi todettuja käytänteitä. Parhaiden käytäntöjen noudattaminen varmistaa, että dokumentaation on laadukasta, saatavilla ja ajan tasalla. Dokumentaation standardointi helpottaa jatkokehitystä ja säästää aikaa (Nayeem Islam 2024.) On kuitenkin huomionarvoista, että vanhentuneellakin dokumentaatiolla on merkitystä ohjelmistokehityksessä.

Dokumentaatiota laadittaessa tulisi pitää mielessä laajuus ja tavoite, mitä sillä halutaan saavuttaa. Kun tavoitteet ovat selviä on helpompi kohdentaa dokumentaatio oleellisimpiin asioihin. Sisällönhallinta on keskeisessä osassa laadukkaan dokumentaation luomisen kannalta. Tämä tarkoittaa dokumentaation päivittämistä, resursointia ja arviointia. Dokumentaatiossa kannattaa suosia yhtenäistä tyyliä ja välttää teknistä kapulakielistä sanastoa, jotta sisältö on helpommin ymmärrettävää. Käytännön esimerkit auttavat hahmottamaan monimutkaisempia kokonaisuuksia ja lisää dokumentaation arvoa. Käyttäjäpalaute hyödyntäminen ja ohjeiden vertaisarviointi ennen julkaisua auttaa dokumentaation jatkokehityksessä (Nayeem Islam 2024.)

4 TIETOJÄRJESTELMÄYMPÄRISTÖ JA KOKONAISARKKITEHTUURI

4.1 IFS - toiminnanohjausjärjestelmä

Eltelillä on käytössä IFS laaja toiminnanohjausjärjestelmä (ERP, Enterprice Resource Planning) jonka tarkoitus on tukea ja hallita yrityksen keskeisiä toimintoja yhdessä tietojärjestelmässä. Järjestelmä kokoaa taloushallinnon, hankinnan, projektinhallinnan, varastonhallinnan ja työnohjausprosessit yhteen kokonaisuuteen mikä mahdollistaa toiminnan tehokkaamman ohjaamisen ja reaaliaikaisen tiedonkulun (IFS Applications 9 Technical Documentation.)

IFS perustuu ERP järjestelmille tyypilliseen keskitettyyn tietokantaan, johon kaikki järjestelmän tuottama data tallentuu (IFS Applications 9 Technical Documentation). Keskitetyn tiedon avulla voidaan vähentää manuaalista tiedon siirtoa, päällekkäistä työtä ja virheiden mahdollisuutta, mikä puolestaan parantaa prosessien läpinäkyvyyttä ja raportoinnin luotettavuutta. Nestell ja Olson (2017) korostavat, että ERP järjestelmien suurimpia hyötyjä ovat prosessien standardointi, datan yhdenmukaisuus sekä mahdollisuus integroida organisaation toimintoja yhdeksi kokonaisuudeksi, mikä vähentää hajanaisten järjestelmien aiheuttamia riskejä ja kustannuksia.

IFS on modulaarinen järjestelmä, mikä tarkoittaa, että sen toiminnot koostuvat erillisistä mutta toisiinsa integroituvista moduuleista. (IFS Applications 9 Technical Documentation). Modulaarisuus mahdollistaa sen, että organisaatiot voivat ottaa käyttöön liiketoiminnan kannalta tarpeelliset kokonaisuudet ja laajentaa järjestelmää vaiheittain toimintaympäristön muuttuessa. Erilaiset moduulit, kuten varastonhallinta, projektinhallinta ja työnohjaus muodostavat loogisen kokonaisuuden, jonka avulla liiketoiminta voi toimia erilaisissa toimintaympäristöissä. (Nestell & Olson 2017, 37-39.)

Tietojärjestelmien merkitys korostuu toimialoilla, joilla toiminta on maantieteellisesti hajautunutta tai joissa työ suoritetaan toimiston ulkopuolella. Eltelin tapauksessa työtä tehdään projektinomaisesti ympäri Suomea, ja työympäristöt vaihtelevat suuresti projektien välillä. Modernit ERP järjestelmät integroidaan usein ulkoisiin mobiilikäyttöliittymiin ja työnohjausjärjestelmiin, mikä mahdollistaa reaaliaikaisen raportoinnin ja seurannan myös hankalissa kenttäolosuhteissa. (Ganeshan, 2020.)

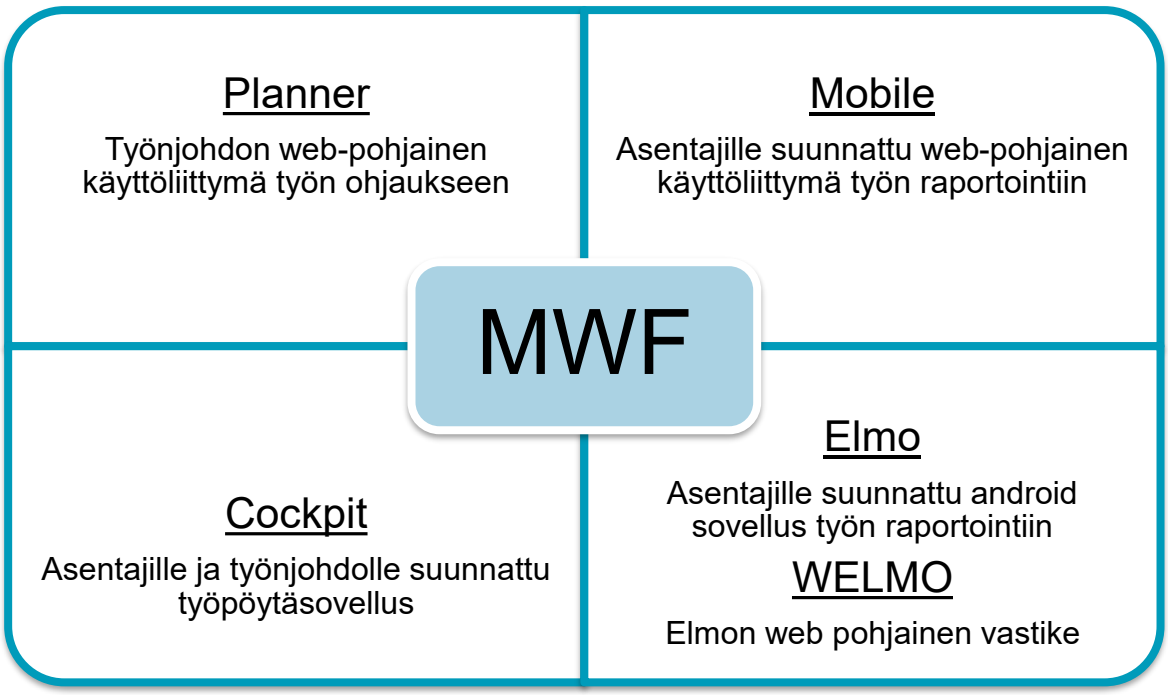
ERP järjestelmän käyttöönotto ei kuitenkaan ole pelkästään tekninen projekti, vaan usein se edellyttää organisaation toimintatapojen tarkastelua ja prosessien yhdenmukaistamista. McCue (2020) ja Hoffman (2024) tuovat esille, että ERP-hankkeiden haasteet liittyvät usein muutoksenhallintaan, prosessien standardointiin ja siihen että organisaation on kyettävä muokkaamaan toimintaansa järjestelmän vaatimusten mukaisiksi. Onnistunut käyttöönotto edellyttää selkeästi määriteltyjä, ja hyvin dokumentoituja prosesseja, sillä järjestelmän tuottama arvo riippuu pitkälti siitä, miten hyvin prosessit on osattu kuvata ja jalkauttaa.

Yhteenvetona voidaan todeta, että IFS on keskeinen osa Eltelin operatiivista toimintaa ja sen vaikutus ulottuu suoraan organisaation prosesseihin ja toimintamalleihin. Ajantasainen dokumentaatio ja prosessien kuvaus tukee liiketoiminnan tavoitteita ja muodostaa perustan myös järjestelmän jatkokehitykselle.

4.2 MWF – työnohjausjärjestelmä

Eltelillä on käytössä ohjelmistotoimittaja isMobilen kehittämä työnohjausjärjestelmä MWF (Mobile Work Force). Se sisältää toiminnallisuuksia työn suunnitteluun, resursointiin, raportointiin sekä projektinhallintaan. MWF sisältää myös low-code-pohjaisen kehitysalustan, jonka avulla järjestelmää voidaan räätälöidä liiketoiminnan tarpeisiin ilman laajamittaista ohjelmistokehitystä. Tämä mahdollistaa ryhmäkohtaiset näkymät, työnkulut sekä prosessit, jotka voivat vaihdella asiakkuuksien ja palvelualueiden välillä (isMobile 2026.)

Kuvassa 5 on esitetty MWF järjestelmän viisi erilaista käyttöliittymää, jotka palvelevat eri rooleja ja käyttötarpeita.



Kuva 5. MWF käyttöliittymät

Kaikkia käyttöliittymiä kehitetään keskitetysti samasta MWF-järjestelmästä käsin (kuva 5), ja käyttäjienhallinta perustuu roolipohjaisuuteen. Roolit määrittelevät mitä näkymiä, toiminnallisuuksia ja tietoja käyttäjä voi tarkastella ja muokata (isMobile 2026.)

MWF kehitystyö toteutetaan isMobilen Blå Studio kehitysympäristössä, joka yhdistää low-code ominaisuudet konfiguroitaviin käyttöliittymäkomponentteihin ja JavaScriptillä toteutettuun logiikkaan. Käyttöliittymien ulkoasu ja rakenne määritellään XML-pohjaisten konfiguraatiotiedostojen avulla, ja kehittäjillä on käytössään valikoima valmiita uudelleenkäytettäviä komponentteja, kuten listoja, painikkeita ja lomakkeita (isMobile 2026.)

Varsinainen sovelluslogiikka rakentuu valmiiden tilojen, siirtymien ja trigger toimintojen ympärille, ja sitä voidaan täydentää ja muokata JavaScriptillä. Vaikka MWF:n low-code kehitys ympäristö vähentää raskasta ohjelmistokehitystä, se edellyttää edelleen perustason ohjelmointiosaamista, jotta järjestelmään voidaan toteuttaa laajempia logiikkamuutoksia tai asiakaskohtaisia toiminnallisuuksia (isMobile 2026.)

Tämän opinnäytetyön lähtökohtana toiminut projektinhallintamalli ohjaa suoraan MWF toiminnallista logiikkaa. Mallinnus määrittää miten prosessien tulee edetä ja mitä asioita tulee raportoida. Siksi prosessien mallinnus ja dokumentointi ovat ratkaisevan tärkeitä työnohjausjärjestelmän kehitystyössä.

4.3 Frends – integraatioalusta

Frends on eurooppalainen integraatioalusta (iPaas, integration Platform as a Service), jota Eltel hyödyntää eri tietojärjestelmien välisen tiedonsiirron ja työnkulkujen hallintaan. Alusta on suunniteltu yhdistämään sovelluksia, tietolähteitä ja räätälöityjä järjestelmiä yhtenäiseksi kokonaisuudeksi ilman että järjestelmien välille tarvitsee rakentaa erillisiä point-to-point integraatioita. Tämä mahdollistaa skaalautuvan ja toimintavarman integraatioarkkitehtuurin, joka tukee organisaation liiketoimintaprosesseja ja reaaliaikaista tiedonkulkua (Junnila 2024.)

Frendsin arkkitehtuuri perustuu prosessipohjaiseen malliin, jossa integraatiot kuvataan selkeinä työnkulkuina. Työnkulut muodostuvat yksittäisitä tehtävistä (task), kuten API-kutsuista, tiedostojen käsittelystä tai datan muuntamisesta, ja ne ketjutetaan haluttuun järjestykseen prosessilogiikan mukaisesti (Frends 2025). Prosessit käynnistyvät triggereiden avulla. Triggeri voi aktivoitua esimerkiksi ajastuksella, ulkoisesta järjestelmästä saapuvasta pyynnöstä, API-kutsusta tai tapahtumasanomasta (task) (Frends 2025.) Näin voidaan automatisoida järjestelmien välisiä tapahtumia ja varmistaa että tieto päivittyy oikea-aikaisesti. Taskit hyödyntävät Frendsin tarjoamia valmiita komponentteja, mutta ne voidaan myös räätälöidä vastaamaan yrityksen yksilöllisiä integraatiotarpeita (Frends 2025).

Alusta tukee yleisiä protokollia ja rajapintateknologioita kuten REST, SOAP, FTP ja tietokantayhteydet, sekä mahdollistavat datan muunnoksen eri formaattien, kuten JSON, XML ja CSV välillä. Datan muuntaminen on keskeistä integraatioissa koska se varmentaa tiedon yhteensopivuuden ja mahdollistaa tiedon muokkaamisen ja rikastamisen (Junnila A 2024, Frends 2025.)

Alustaan sisältyy myös hallintanäkymä, josta integraatioiden tilaa voidaan valvoa reaaliaikaisesti. Valvonta ja lokitiedot tukevat prosessien läpinäkyvyyttä ja auttavat havaitsemaan virheitä ja poikkeamia. Lisäksi Frends tarjoaa sisäänrakennetut virheenkäsittelyominaisuudet ja mahdollisuuden automatisoida hälytyksiä, mikä parantaa tiedonkulun luotettavuutta projektien ja työtilausten osalta (Frends 2025.)

Eltelin ympäristössä Frends yhdistää keskeiset järjestelmät ja varmistaa että tiedonsiirto tapahtuu prosessien mukaisesti. Tämän vuoksi integraatioalustan rooli on tärkeä myös tämän opinnäytetyön kannalta.

5 TYÖN TOTEUTUS JA KEHITYSPROSESSI

5.1 Tutkimusote

Opinnäytetyö toteutettiin tutkimuksellisena kehitystyönä, joka lähti liikkeelle käytännön tarpeesta parantaa projektinhallinnan tukea työnohjausjärjestelmässä sekä kehittää yhtenäisempiä ja paremmin ylläpidettäviä dokumentointikäytäntöjä. Tutkimuksellinen kehitystyö sisältää vaiheita ja menetelmiä kuten nykytilan sekä tahtotilan analysointia, ongelmakohtien tunnistamista ja vaihtoehtoisten ratkaisujen arviointia, joita myös hyödynnettiin tässä opinnäytetyössä.

Projektin hallinnassa sovellettiin ketteriä menetelmiä, mikä tarkoittaa iteratiivista ja joustavaa lähestymistapaa kehittämiseen. Suunnittelu–toteutus–arviointi sykli (Ojasalo ym. 205, 23) toistui tiheästi projektin aikana, mikä mahdollisti nopean reagoinnin muutoksiin ja jatkuvan parantamisen projektin aikana. Tämä tuki kehittämistyön tavoitteita ja varmisti, että ratkaisu vastasi käytännön tarpeisiin mahdollisimman hyvin.

5.2 Menetelmät ja aineiston keruu

Ojasalon ym. (2025) ja Vilkka (2017) mukaan tutkimuksellisessa kehittämistyössä korostuu usein ideoiden ja ratkaisujen mallintaminen, kuvaaminen sekä käyttöönotto käytännön tasolla. Tässä opinnäytetyössä aineistoa kerättiin havainnoinnin, haastatteluiden ja dokumenttianalyysin avulla. Haastattelut asiantuntijoiden ja työnjohdon kanssa toivat esiin käytännön haasteita, hiljaista tietoa sekä tarpeen selkeyttää dokumentointikäytäntöjä. Havainnoinnin ja aiemman projektimateriaalin perusteella muodostui kuva nykyisistä prosesseista, järjestelmien välisistä riippuvuuksista ja kehitystarpeista.

Dokumenttianalyysin osalta perehdyin aikaisempien projektien dokumentaatioon, jota oli saatavilla melko niukasti. Käytettävissä ollut prosessikaavio tarjosi tärkeän lähtökohdan opinnäytetyöhön. Lisäksi aineistona hyödynnettiin asiakkaan tietojärjestelmää ja aiemmin määriteltyjä asiakasvaatimuksia. Aineisto osoitti, että dokumentaation suhteen kehitykseltä puuttui yhdenmukaiset toimintatavat, eikä dokumentaatiolle ollut selkeitä vaatimuksia. Tämän vuoksi dokumentoinnin parantaminen nousi luontevasti yhdeksi opinnäytetyön keskeisimmistä osa-alueista.

Tutkimuksellisuus ilmeni kehitystyössä järjestelmällisenä etenemisenä, uuden tiedon tuottamisena sekä analyyttisenä otteena. Kehittämisen tavoitteena ei ollut ainoastaan ratkaista käytännön ongelmaa, vaan myös luoda dokumentoitu ja perusteltu malli, joka tukee organisaation pitkäjänteistä kehittämistä (Ojasalo ym., 2015, 19–20.)

5.3 Lähtötilanne ja suunnittelu

Eltelillä on meneillään suuria projekteja tietojärjestelmien uudistamiseksi. Kehitysorganisaatio muodostuu Business IT yksiköstä ja liiketoiminnan asiantuntijoista. Käytännön toteutuksesta vastaa pienikokoinen ydin tiimi, johon kuuluu ohjelmistokehittäjä, integraatioasiantuntija, liiketoiminnan edustaja ja kehityspäällikkö. Oma roolini kehityskoordinaattorina on toimia linkkinä IT:n ja liiketoiminnon välillä: tunnistaa ja edistää kehityskohteita, tukea loppukäyttäjiä sekä varmistaa että kehittämistä ohjaava tieto on kootusti saatavilla.

Järjestelmäkehitys etenee nopeasti, tämän takia dokumentaatio on jäänyt monelta osin kehityksen jalkoihin. Kehitystä tehdään integraation ja työnohjausjärjestelmän osalta low-code työkaluja hyödyntäen ja tarvittaessa neuvotellaan järjestelmätoimittajan kanssa isommista muutoksista. Opinnäytetyön aikana esiin nousi tarve parantaa dokumentointikäytäntöjä sekä kuvata prosesseja liiketoiminnan kehittämisen näkökulmasta. Tavoitteena oli tuottaa käytännöllinen, kevyt ja ylläpidettävä dokumentointimalli sekä koota asiakaskohtainen kokonaiskuva, joka tukee kehitystä.

Kehitysprojektien koordinointi ja hallinta aloitettiin Microsoft Plannerin avulla. Kehitysympäristöön muodostettiin suurempia kehityskokonaisuuksia, jotka sitten pilkottiin tarvittaessa pienemmiksi tehtäviksi. Kehitysympäristöä ylläpidettiin Teamsissa, jonne myös kerättiin aineistoa kehitysprojekteista ja kommentoitiin tilannetta viikoittain tehtäväkorteille sekä palaverimuistioihin. Tällä oli merkittävä osuus opinnäytetyön aineiston keruun näkökulmasta. Myöhemmässä vaiheessa kehitysympäristö siirrettiin Sharepointiin, jossa tehtäviä muokattiin pienemmiksi, jolloin myös kehitysprojektien seurattavuus sekä ymmärrys kehityksen eri vaiheista parani myös liiketoiminnan näkökulmasta. Erityisen tärkeää oli saada dokumentoitua esteet ja viivästykset, joita kehityksen aikana ilmaantui.

Eltelillä on suurimpien asiakkaiden kanssa yhteinen ohjelmistorajapinta, jonka kautta työtilauksia käsitellään. Tämä mahdollistaa saman järjestelmän käytön kaikille roolista riippumatta, jolloin oma toimintamalli säilyy ja sitä voidaan kehittää. Tämä mahdollistaan myös datan keräämisen ja rikastamisen sekä automaation käytön työn raportoinnissa. Data voidaan tarvittaessa myös validoida ennen lähetystä, jolloin laatu paranee ja mahdollisia virheitä voidaan vähentää. Lisäksi rajapinta tarjoaa paremman näkyvyyden omien töiden tilaan ja valmiusasteisiin, mikä on erityisen arvokasta työnjohtamisen ja raportoinnin kannalta. Frends alustan avulla valmiita rajapinta prosesseja voidaan hyödyntää uudelleen eri asiakkuuksilla, mikä nopeuttaa kehitystä ja ylläpitoa.

Töitä edistetään ja raportoidaan oman MWF työnohjausjärjestelmän avulla ja tietoa viedään ja tuodaan rajapinnan yli aivan kuten sitä käsiteltäisiin suoraan asiakkaan omassa järjestelmässä. Jokaisella asiakkuudella on omat vaatimukset ja yhteisesti sovitut säännöt, joiden mukaan töitä edistetään ja raportoidaan. Tämän opinnäytetyön lähtökohtana oli asiakasvaatimusten perusteella tehty työnkulun prosessi sekä muut asiakkuuden dokumentit.

Mallinnuksella on suuri rooli järjestelmäkehityksen kannalta koska se tulisi ohjaamaan kehitystä sekä integraation että työnohjausjärjestelmän osalta. Nykyisestä prosessista (kuva 6) tunnistettiin useita asioita, joihin haluttiin parannusta. Keskeisin haaste liittyi työtilausten käsittelyyn. Työtilauksia käsiteltiin tietojärjestelmissä yhdellä prosessilla, jossa työtä edistettiin kronologisesti vaiheesta toiseen, vaikka tosiasiassa samaan työtilaukseen kohdistui useita rinnakkaisia työvaiheita samaan aikaan. Tämän seurauksena kaikki tulot ja kulut kohdistuvat yhdelle työnumerolle riippumatta siitä mitä työvaihetta on tehty.

Kuva 6. Tunnistetut nykyisen prosessin ongelmakohdat

Raportointi ja talouden seuranta oli hankalaa koska erittelyä piti tehdä käsin esimerkiksi sen selvittämiseksi, liittyivätkö laskut maanrakennukseen vai asennukseen. Tarvittiin siis selkeämpää rakennetta töiden raportointiin ja kulujen kohdistamiseen töiden eri vaiheissa.

Toinen merkittävä ongelma liittyi resursointiin. Työnohjausjärjestelmä salli työtehtävän osoittamisen vain yhdelle resurssille kerrallaan. Tämä vääristi resurssienhallintaa koska järjestelmä näki asentajan vapaana, vaikka se tosiasiassa oli varattuna toiselle työmaalle. Järjestelmän jäykkyys heikensi siis työnjohdon näkymää resursoinnin suhteen.

Tämä ja muut käytettävyysongelmat johtivat siihen, että johtamista ja raportointia tehtiin järjestelmien ulkopuolella erilaisten Excel taulukoiden avulla. Kentällä oli siis muodostunut erilaisia alueellisia ja henkilökohtaisia toimintatapoja, jotka heikensivät yhtenäistä ja standardoitua raportointia ja työnseurantaa.

5.4 Tavoitetila ja ratkaisut

Työnohjausjärjestelmää oli kehitettävä siten, että projekteja voitaisiin hallita, johtaa ja raportoida riittävän hyvin järjestelmästä käsin, ilman tarvetta ulkopuolisille Excel taulukoille. Tavoitteena oli luoda yhtenäinen toimintamalli, joka tuottaa reaaliaikaista tietoa projektien taloudellisesta ja tuotannollisesta tilanteesta. Tämä parantaa läpinäkyvyyttä ja mahdollistaa tiedolla johtamisen, kun operatiivinen johto voi tehdä päätöksiä ajantasaisen tiedon perusteella (kuva 7).

Kuva 7. Tavoitetila ja hyötynäkökulmat

Mallinnuksessa käytettiin Microsoft Visio työkalua ja uimarata kaaviota, jossa näkyy eri järjestelmien roolit sekä vaiheet, jotka prosessin aikana suoritetaan. Samantyyppistä mallia oli käytetty myös aikaisemmissa projekteissa ja se oli todettu toimivaksi.

Mallinnuksen lähtökohtana oli, että työtilauksia pitäisi pystyä pilkkomaan pienempiin osiin, jolloin työn hallinta ja seuranta helpottuu. Työtilauksesta muodostetiin aliprosesseja kuten asennus ja maanrakennus, jotka sitten voitiin antaa tehtäväksi asentajille, suunnittelijoille tai maanrakentajille tarpeen mukaan. Mallinnus vaiheessa tarkasteltiin erityisesti kahta näkökulmaa, mitä tehtäviä kullekin työtilaukselle kuuluisi muodostua ja missä järjestyksessä sekä sitä, miten aliprosesseja tulisi edistää suhteessa pääprosessiin. Tämä tarkentui prosessin aikana, kun testaamista tehtiin integraation kanssa. Laajensin kuvausta useammalle tasolle, jolloin oli mahdollista kuvata yksittäisen prosessin sijaan myös asiakkuuteen liittyvät muut prosessit ja aliprosessit. Tästä syntyi eräänlainen asiakkuuskortti, josta selviää asiakkuuden keskeisimmät vaatimukset sekä järjestelmien että liiketoiminnan kannalta.

Prosessi kuvattiin sekä ylätasolta että aliprosessien osalta siltä osin kuin tietoa oli saatavilla. Tarkoituksena oli muodostaa kokonaiskuva ja parantaa prosessien ymmärrettävyyttä myös aliprosessien osalta. Ylätason kuvaukseen on liitetty myös muita kuin tämän opinnäytetyön kannalta oleellisia asi-

oita. Tietovirtoja kuvattiin ylätasolta siten että järjestelmien keskinäiset riippuvuudet selvisivät riittävällä tarkkuudella. Frends integraatioalustalta on tarvittaessa saatavissa tarkka dokumentaatio tietovirroista, joten en nähnyt tarpeelliseksi kuvata sitä kovin tarkasti näihin dokumentteihin.

Muutos tehtiin sisäisiin prosesseihin mikä tarkoittaa, että muutos ei vaikuttanut asiakkaaseen eikä asiakasrajapintaan tarvinnut tehdä muutoksia. Työtilauksia käsiteltäisiin omissa järjestelmissä jatkossa uuden prosessin mukaisesti. Tämä tarkoitti luonnollisesti myös sitä, että käyttäjille piti tarjota ohjeistus ja koulutusta uuden toimintamallin suhteen.

Dokumentaatiokäytänteitä oli tarkasteltava erityisesti ketterän kehityksen näkökulmasta siten että liiketoiminta saa riittävästi tietoa kehityksen johtamiseen, jatkokehittämiseen sekä datanhallintaan ja säilyttämiseen liittyvistä asioista. Dokumentaation tulee tukea myös järjestelmien keskinäisten riippuvuuksien ymmärtämistä, jotta kokonaisuuden hallinta on mahdollista. Samalla dokumentoinnin on oltava riittävän kevyttä, että se ei hidastaisi kehitystä merkittävästi. Myös uudet ominaisuudet sekä ohjelmiston toimintalogiikka pitäisi pystyä dokumentoimaan entistä paremmin, jolloin myös jatkokehitys tehostuu, kun edellisien projektien dokumentaatio on paremmin saatavilla tulevien projektien tueksi.

Oli selvää, että perinteinen vaatimusmäärittely ei toimisi tässä ympäristössä sillä muutoksia syntyy paljon ja etukäteen laadittava raskas dokumentaatio myös hidastaisi kehitystä tarpeettomasti. Dokumentoinnin tulisi sen sijaan tukea suunnittelua, toteutusta ja testausta riittävässä laajuudessa. Siitä pitäisi pystyä tunnistamaan kehityksen keskeiset vaiheet, havaitut ongelmakohdat, tavoitetila sekä projektin aikana tehdyt muutokset ja uudet ominaisuudet.

Myös testausvaiheeseen tarvittiin selkeämpiä käytäntöjä testauksen kattavuuden ja valmiin työn määrittämiseksi. Kehitystyön vaiheistusta selkeytti se, että kehitysympäristön hallinta siirrettiin Sharepointiin. Siirtymän yhteydessä kehitysprosessi jaettiin selkeisiin vaiheisiin; suunnittelu, rakentaminen, testaaminen ja jalkautus. Tämä toi kaivattua rakennetta sekä kehityksen johtamiseen että dokumentaation muodostamiseen.

Olin tehnyt valmiiksi vaatimusmäärittelyn ja projektidokumentaatiopohjan ensimmäisen version, jonka kuitenkin hylkäsin, kun huomasin että se ei vastannut kehityksen tarpeita. Vaatimuksia oli hankalaa tunnistaa etukäteen ja niitä tuli koko ajan lisää projektin edetessä. Aloin lisätä tehtäviä ja yksittäisiä testitapauksia Exceliin, että pysyisin paremmin perillä mitä oli tehty toteutuksen ja testaamisen suhteen. Tästä alkoi muodostua elävä dokumentti (liite 1), johon lisäsin tehtäviä sekä testitapauksia projektin edetessä. Ajatus tällaisesta dokumentointimallista syntyi perehdyttyäni ketteriin menetelmiin ja niissä hyödynnettyihin käyttäjä tarinoihin. Sovelsin samaa periaatetta siten, että tarinan sijasta kuvasin tehtävän joka järjestelmän tai käyttäjän pitäisi pystyä suorittamaan ja lisäsin siihen hyväksymiskriteerin. Näin syntynyt dokumentti toimi samanaikaisesti sekä kevyenä vaatimusmäärittelynä, testidokumentaationa sekä kehitystä ohjaavana työkaluna. Samasta dokumentista voitiin muodostaa tehtävät kehitysympäristöön, ja se kuvasi selkeästi projektin tavoitteet, kehitysvaiheet sekä mahdolliset ongelmakohdat. Dokumenttia tuli päivittää projektin aikana aivan viimeiseen vaiheeseen saakka, jolloin se ei jäänyt irralliseksi osaksi kehitystä vaan toimi kehitystä ohjaavana tekijänä. Dokumentaatiota ja mallinnusta tuli siis tehdä yhtäaikaisesti läpi projektin. Tällä tavoin syntyi sekä yksityiskohtainen kuvaus toimintalogiikasta, kattava testidokumentaatio sekä ylätason prosessikuvaus, joka tuki käyttöönottoa sekä tulevia kehityshankkeita.

Käyttöohjeet ja muu koulutusta tukeva dokumentaatio tulee tehdä vasta projektin loppuvaiheessa, kun kaikki toiminnot ovat vakiintuneet ja muutoksia ei enää tehdä. Näin toimin myös projektimoduulin käyttöohjeen kanssa. Vaikka muutoksen tuomat hyödyt ovatkin suuret, ohjelmiston käyttäminen ei käyttöliittymätasolla muuttunut kovinkaan paljoa, joten tämä osuus oli suhteellisen pieni. Muutokset tehtiin olemassa olevan pikaohjeeseen, johon kuvasin uuden prosessin kannalta oleelliset toiminnot kuten resursointiin ja vastuuttamiseen liittyvät asiat.

5.5 Projektinhallintamallin toteutus työnohjausjärjestelmään

Talouden paremman seurattavuuden vuoksi työtilauksille piti luoda oikeanlainen rakenne IFS toiminnanohjausjärjestelmään. Rakenne oli suunniteltu yhdessä projektipäälliköiden ja operatiivisen johdon kanssa ja IFS tiimi hoiti teknisen toteutuksen. Projektirakenteessa oli huomioitu kustannukset, joita työtilauksille voisi muodostua.

Kun projektirakenne oli luotu, voitiin työstä muodostaa IFS järjestelmään tarvittava kokonaisuus ja kytkeä uusi työtilaus talousprojektille. Tämän jälkeen integraation oli luotava alustavan prosessimallin mukaisesti työtilaus sekä aktiviteetit MWF järjestelmään oikeassa järjestyksessä. Tässä oli huomioitava useita asioita kuten työntyyppi sekä alityyppi. Vain tietyntyyppiset tehtävät, joihin oli otettu kantaa prosessimallinnuksessa, tulisi kytkeä projektirakenteeseen. Kaikkia muita työtyyppejä edistettäisiin yhdellä prosessilla kuten aikaisemminkin. Projektirakenteen perustella muodostettiin työnohjausjärjestelmään aliprosessitaso eli aktiviteetit, joita sitten voitiin antaa asentajille ja suunnittelijoille tehtäväksi.

MWF järjestelmään piti luoda rinnakkainen prosessi, jota voitiin edistää pääprosessin rinnalla. Nämä prosessit tuli myös kytkeä toisiinsa työtilauksen perusteella. MWF järjestelmässä luotiin erilliset työnkulut aliprosessille ja pääprosessille, jolloin molempia voitiin muokata erillisinä osakokonaisuuksina. MWF low-code alusta mahdollisti aiempien prosessien hyödyntämisen, jolloin kaikkea toiminnallisuutta ei tarvinnut tehdä alusta alkaen. Projektin alussa tunnistettiin mahdolliset erot aiempiin prosesseihin, jotka piti räätälöidä vastaamaan tämän asiakkuuden vaatimuksia.

6 YHTEENVETO JA TULOKSET

Opinnäytetyön tavoitteena oli parantaa ja lisätä kehitysprojektien aikaista dokumentaatiota sekä laajentaa prosessien kuvaamisen mallia Eltelin tietojärjestelmissä. Perimmäisenä ajatuksena oli lisätä kehitysprojektien seurattavuutta ja läpinäkyvyyttä myös liiketoiminnon kannalta. Opinnäytetyö tarkoituksena oli myös tukea suurempaa kehitysprojektia työnohjausjärjestelmän parantamiseksi sekä tulevia kehityshankkeita.

Opinnäytetyön tuloksena syntyi laajennettu malli Eltelin järjestelmien prosessien kuvaamiseen. Mallin avulla voidaan aiempaa paremmin kuvata asiakkuuteen liittyviä prosesseja sekä aliprosesseja. Tämä tukee sekä business it:n että liiketoiminnan tavoitteita, kehitystyötä ja tulevien kehitysprojektien suunnittelua.

Lisäksi opinnäytetyössä kehitettiin uudenlainen mallin kehitystyön aikaiseen dokumentointiin. Malli tuki vaatimustenmäärittelyä sekä testaamista ketterässä ja nopeatempoisessa ympäristössä ja auttoi tunnistamaan kehityksen pullonkauloja sekä järjestelmälogiikkaa ja niiden välisiä riippuvuuksia ja riskejä.

Kehitysympäristön osalta siirryimme Plannerista Sharepointiin, jonne saimme luotua rakenteisemman kehitysympäristön, joka auttoi luomaan paremman tavan kuvata tehtäviä ja seurata kehitystä eri kehitysvaiheiden kautta.

Johtopäätöksenä voidaan sanoa, että uusi malli, kehitysympäristö ja dokumentaatiopohja auttoivat saamaan paremman kuvan kehityksen vaiheista ja ongelmakohdista. Se lisäsi läpinäkyvyyttä ja auttoi hahmottamaan paremmin järjestelmien välisiä riippuvuuksia. Tämä loi paremman pohjan tuleville kehityshankkeille, ja oikein toteutettuna tulisi vähentämään tiedon siiloutumista organisaation eri liiketoimintojen välillä.

7 POHDINTA

Tämä opinnäytetyö auttoi merkittävästi nykyisen roolini haltuunottoa ja hahmottamaan kehityskohteita laajemmin dokumentaation ja kehityksen näkökulmasta. Tutkittuani aikaisempia tutkimuksia ja alan kirjallisuutta ymmärsin paremmin dokumentoinnin merkityksen IT kehityksen tukena ja tulevien projektien ja hiljaisen tiedon siirtämisen kannalta. Teoria, havainnointi ja käytännön työtehtävät ovat lisänneet ymmärrystäni projektinhallinnan ja ketterien menetelmien soveltamisen osalta.

Tiedon jakamisen ja päivittäisen kommunikoinnin tärkeyttä ei voi liikaa korostaa. Tämä on erityisen tärkeää ympäristössä, jossa työntekijät ovat maantieteellisesti hajallaan. Tiimien on muodostettava hyviä käytäntöjä ja kanavia tiedon jakamiseen siten, että sidosryhmät ja tiiminjäsenet pysyvät tilanteen tasalla. Tämä edistää myös kehityksen sujuvuutta ja kehityksen pullonkaulojen tunnistamista. Tähän pitäisi jatkossa kiinnittää enemmän huomiota.

Opinnäytetyön tulosten jalkauttaminen ja käytäntöjen muuttaminen dokumentaation osalta tulee olemaan seuraava askel. Dokumentaatiopohja tulisi ottaa laajemmin käyttöön, testata ja kehittää sitä siten että sen käyttäminen ei muodostu kehityksen hidasteeksi. Tehtävien kokoluokan määrittely tulee tehdä huolellisesti, ettei dokumentaatio laajene liikaa ja käy liian raskaaksi. Lisäksi myös muista asiakkuuksista tulisi tehdä laajennettu prosessimalli, että asiakkuuksien väliset erot prosesseissa selviää.

Prosesseja tulisi kuvata enemmän ja laajemmin, ja ohjeistusta tulee lisätä esimerkiksi uusien asiakkuuksien sekä rajapintojen osalta. Käyttöohjeiden osalta tulee määrittää vakiintunut käytäntö, omistajuus ja vastuut sekä sisällyttää niihin versionhallinta. Ohjeet tulee myös päivittää säännöllisesti ja ne on tehtävä vasta, kun ominaisuudet ja toiminnot ovat vakiintuneet.

Jatkokehitystä ajatellen kokonaisuutena tärkeintä olisi keskittyä muodostamaan vakiintuneita käytäntöjä dokumentoinnin suhteen sekä lisätä kuvauksia eri prosesseista ja teknisistä ratkaisuista sekä kiinnittää huomiota siihen että tieto liikkuu sidosryhmien välillä riittävästi.

LÄHTEET

Työssä on käytetty tekoälyä seuraavasti: M365 copilot. OpenAI. GPT-4.0. Käytetty kielentarkistukseen, tekstin rakenteen muokkaamiseen sekä hakusanojen tuottamiseen tiedonhakua varten, helmikuu 2026.

Adetokunbo A.A. Adenowo, Basirat A. Adenowo 2013. Software Engineering methodologies: a review of the waterfall model and object-oriented approach. International Journal of Scientific & Engineering Research 4.7 (2013): 427-434. Verkkojulkaisu. https://www.researchgate.net/profile/Adetokunbo_Adenowo/publication/344194737_Software_Engineering_Methodologies_A_Review_of_the_Waterfall_Model_and_Object-_Oriented_Approach/links/5f5a803292851c07895d2ce8/Software-Engineering-Methodologies-A-Review-of-the-Waterfall-Model-and-Object-Oriented-Approach.pdf. Viitattu 14.12.2025

Artto K., Martinsuo M., Kujala J. Projektiliiketoiminta. Verkkokirja. Tuotanto talouden laitos. Aalto yliopisto. https://www.aalto.fi/sites/default/files/2020-08/Projektiliiketoiminta.pdf. Viitattu 15.11.2025

Beck. K, Beedle. M, Bennekum. Are Van, Cockburn. A, Cunningham. W, Fowler. M, Grenning. J, Higsmith. J, Hunt. A, Jeffries. R, Kern. J, Marick. B, Martin. R, Mellor. S, Schwaber. K, Sutherland. J, Thomas D. Agilemanifesto.org 2001. Verkkojulkaisu. Viitattu 16.11.12025.

de Souza, S.C.B., Anquetil, N. & de Oliveira, K.M. Which documentation for software maintenance?. J Braz Comp Soc 12, 31–44 2006. https://doi.org/10.1007/BF03194494. Viitattu 15.2.2026

Frends 2025. Connectors. Frends dokumentaatio. Verkkojulkaisu. https://docs.frends.com/frends-development/integrations/connectors. Viitattu 23.11.2025

Ganeshan A. 2020. Field Service Management using ServiceMax. irjet.net. 5.5.2020. Verkkojulkaisu. https://www.irjet.net/archives/V7/i5/IRJET-V7I5675.pdf. Viitattu 19.11.2025

Hanna Vilkka. 2025. Tutki ja kehitä. Santalahti kustannus.

Hoffman S. 2024. The History and evolution of ERP Systems: The Past and Fututure. Software Connect. Verkkojulkaisu. Päivitetty 4.10.2024. https://softwareconnect.com/learn/erp-history/. Viitattu 16.11.2025.

IFS nd. IFS tekninen dokumentaatio. Verkkojulkaisu. https://dixiprod01.ifscloud.net:58443/ifs-doc/f1doc/default.htm. Viitattu 9.3.2026.

isMobile 2026. Verkkojulkaisu. https://ismobile.atlassian.net/wiki/spaces/PLANNER/pages/39715242/Configuration. Viitattu 9.3.2026.

Jeff Sutherland, 2011. Takeuchi and Nonaka: The roots of Scrum. Scrum.inc verkkosivuston blogi. 22.10.2011. https://www.scruminc.com/takeuchi-and-nonaka-roots-of-scrum/. Viitattu 15.11.2025.

Jeston, J. & Nelis, J. 2014. *Business process management: Practical guidelines to successful implementations*. 3rd ed. Abingdon, Oxon:New York: Routledge.VLeBooks. Viitattu 15.2.2026

Junnila A 2024. What is iPaas? Overview of Integration platform as a Service. Frends verkkosivusto blogi. 18.11.2024. https://frends.com/ipaas/blog/what-is-ipaas-integration-platform-as-a-service-explained. Viitattu 16.11.2025

Ken Schwaber and Jeff Sutherland 2020. The Srum Guide. Verkkojulkaisu. https://scrum-guides.org/scrum-guide.html. Viitattu 20.12.2025

Koulutus.fi. 2025. Mitä ovat ketterät menetelmät? – Scrum, Lean ja muut tutuksi. Koulutus.fi verkkosivusto oppaat. Päivitetty 18.8.2025. https://www.koulutus.fi/oppaat/projektinhallinta/ketteratmenetelmat-19939. Viitattu 12.11.2025

Laine 2021. Havaintoja ketteryydestä ja ohjelmistojärjestelmien dokumentaatiosta. Digia Blogi. https://digia.com/blogi/havaintoja-ketteryydesta-ja-ohjelmistojarjestelmien-dokumentaatiosta. Viitattu 15.2.2026.

Layton, Mark C, Ostermiller, SJ, & Kynaston, DJ 2020. Agile Project Management for Dummies. E-kirja. John Wiley & Sons, Incorporated, Newark. ProQuest Ebook Central. Viitattu 15.2 2026.

Lean/Agile Documentation: Strategies for Agile Teams. Verkkosivusto Agile modeling. https://agilemodeling.com/essays/agileDocumentation.htm#WhenIsADocumentAgile. Viitattu 14.2.2026

Martinsuo, M & Blomqvist, M 2010, Prosessien mallintaminen osana toiminnan kehittämistä. Tampereen teknillinen yliopisto. Teknis-taloudellinen tiedekunta. Opetusmoniste, Vuosikerta 2, Tampere. https://cris.tuni.fi/ws/portalfiles/portal/2098668/prosessien_mallintaminen.pdf. Viitattu 31.1.2026

MCCue I. 2020. The History of ERP. Oracle. Päivitetty 12.8.2020. Verkkojulkaisu. https://www.netsuite.com/portal/resource/articles/erp/erp-history.shtml

Microsoft nd. Visio. Verkkojulkaisu. https://www.microsoft.com/fi-fi/microsoft-365/visio/?msockid=220fdc2b24d46c3c38b1ca5925766de0. Viitattu 14.2.2026.

Microsoft nd. Visualisoi liiketoimintaprosessit selkeästi. Verkkojulkaisu. https://www.microsoft.com/fi-fi/microsoft-365/visio/business-process-modeling-notation. Viitattu 14.2.2026

Mikä on ohjelmointirajapinta? nd. Verkkojulkaisu. https://www.sap.com/finland/products/technology-platform/integration-suite/what-is-api.html. Viitattu 6.1.2026

Mäntyneva, M. 2025. Hallittu projekti: Jäntevästä suunnittelusta menestykselliseen toteutukseen. 3., uudistettu painos. Helsinki: Kauppakamari. https://www.kauppakamarikauppa.fi/products/e-kirja-hallittu-projekti-jantevasta-suunnittelusta-menestykselliseen-toteutukseen-3-painos. Viitattu 15.2.2026

Nayeem I. 2024. Mastering The Art of Softwate Documentation: A Comprehensive Guide for Developers and Tech Professionals. Medium blogi. https://medium.com/@nomannayeem/mastering-the-art-of-software-documentation-a06aa5d7e697. Viitattu 15.2.2026

Nestell, JG, & Olson, DL 2017, Successful ERP Systems: A Guide for Businesses and Executives, Business Expert Press. E-kirja. New York. ProQuest Ebook Central. Viitattu 14.2.2026

Object Management Group nd. Graphical notations for business processes. Omg.org verkkojulkaisu https://www.omg.org/bpmn/. Viitattu 28.12.2025

Object Management Group nd. Introduction to Unified Modeling Language (UML). Omg.org verkkojulkaisu. https://www.omg.org/uml/. Viitattu 28.12.2025

Ojasalo K, Moilanen T, Ritalahti J. 2015. Kehittämistyön menetelmät: Uudenlaista osaamista liiketoimintaan. E-Kirja. Helsinki: Sanoma Pro Oy. Viitattu 21.11.2025

Poppendieck M, Poppendieck T. 2013. Lean software development, An agile toolkit. https://ptgmedia.pearsoncmg.com/images/9780321150783/samplepages/0321150783.pdf. Viitattu 15.2.2026

Project Management Institute 2017. Agile Practice Guide. Newtown Squar, PA:Project Management Institute. E-kirja. ProQuest Ebook Central. https://www.pmi.org/pmbok-guide-standards/practice-guides/agile.Viitattu 14.12.2025

Projektinhallinnan perusteet: Kattava opas onnistuneeseen projektinjohtamiseen. 2025. Tieturin blogi. 10.9.2025. https://www.tieturi.fi/blogi/projektinhallinnan-perusteet-kattava-opas-onnistuneeseen-projektinjohtamiseen/. Viitattu 10.11.2025.

Rokis, K. and Kirikova, M. 2023. Exploring low-code development: A comprehensive literature review, *Computer Science and Information Management Quarterly*, 200(36), 68–86. Verkkojulkaisu. https://doi.org/10.7250/csimq.2023-36.04. Viitattu 18.1.2026

Saravanos, A. (2025) A Brief History of the Waterfall Model: Past, Present, and Future. Verkkojulkaisu. https://archive.nyu.edu/bitstream/2451/75504/2/Brief_History.pdf. Viitattu 21.1.2025

Satish, CJ, Anand, M., 2016. Software documentation Management Issues and Practices: a Survey. Indian Journal of Science and Technology, 9(20). Verkkojulkaisu. DOI:10.17485/ijst/2016/v9i20/86869

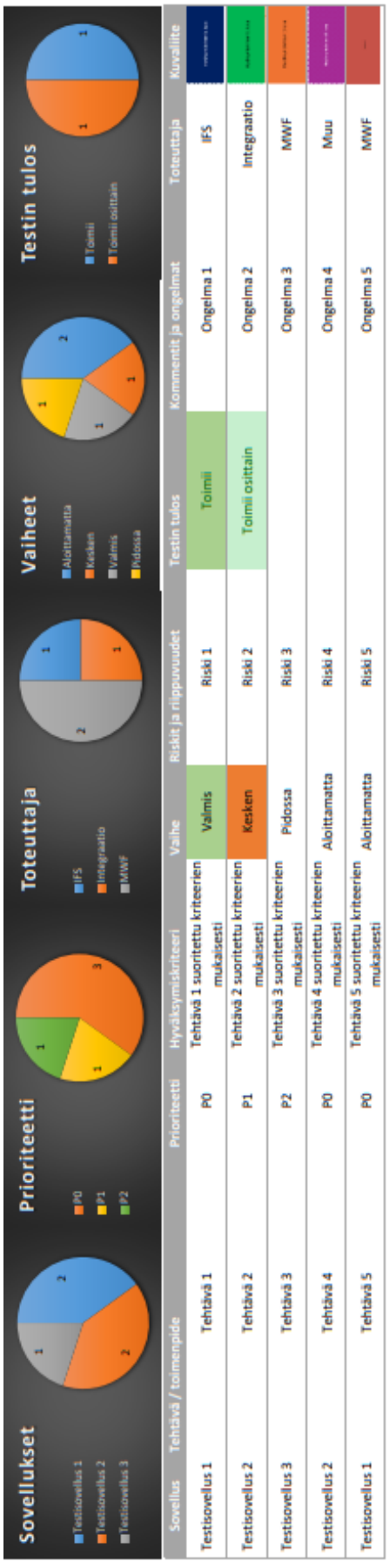
S. W. Ambler. Agile documentation nd. Verkkojulkaisu. http://www.agilemodeling.com/essays/agileDocumentation.htm, Viitattu 11.1.2026

The Standish Group 2015. Chaos report 2015. Verkkosivusto. https://cdn1-public.info-tech.com/agile/CHAOSReport2015-Final.pdf. Viitattu 31.1.2026

Tietoa Eltelistä nd. Verkkojulkaisu. https://www.eltelnetworks.fi/fi/tietoa-eltelista/

Voigt s, Garrel J, Muller J, Wirth D. 2016. A Study of Documentation in Agile Software Projects. Verkkojulkaisu. https://doi.org/10.1145/2961111.2962616

LIITE 1: DOKUMENTAATIOPOHJA



Sovellus	Tehtävä / toimenpide	Prioriteetti	Hyväksymiskriteeri	Vaihe	Riskit ja riippuvuudet	Testin tulos	Kommentit ja ongelmat	Toteuttaja	Kuvaliite
Testisovellus 1	Tehtävä 1	P0	Tehtävä 1 suoritettu kriteerien mukaisesti	Valmis	Riski 1	Toimii	Ongelma 1	IFS	
Testisovellus 2	Tehtävä 2	P1	Tehtävä 2 suoritettu kriteerien mukaisesti	Kesken	Riski 2	Toimii osittain	Ongelma 2	Integraatio	
Testisovellus 3	Tehtävä 3	P2	Tehtävä 3 suoritettu kriteerien mukaisesti	Pidossa	Riski 3		Ongelma 3	MWF	
Testisovellus 2	Tehtävä 4	P0	Tehtävä 4 suoritettu kriteerien mukaisesti	Aloittamatta	Riski 4		Ongelma 4	Muu	
Testisovellus 1	Tehtävä 5	P0	Tehtävä 5 suoritettu kriteerien mukaisesti	Aloittamatta	Riski 5		Ongelma 5	MWF	